Patent No. 3168580

Title: Page scribe language interpreter


Abstract

An information processing system with the following characteristics: In an information processing system embodied by crudely coupling a group of information processing machines inclusive of multiple information processing machines endowed each with a rastering function of obtaining pixel information for printing from a source file, a given information processing machine within the aforementioned group of information processing machines possesses a division mechanism designed to divide the aforementioned source file or an intermediate code file obtained as a result of the conversion of said source file into multiple partial files which can be independently rasterized and a transfer mechanism designed to transfer at least one of the aforementioned multiple partial files into another information processing machine capable of executing a rasterizing routine.

An information processing system wherein the information processing machine that possesses the aforementioned division mechanism and transfer mechanism is also endowed with the aforementioned rasterizing function and enables the procurement of pixel information by restoring the remainder of the aforementioned multiple partial files other than the aforementioned partial file(s) transmitted to the other information processing machine.

An information processing system wherein the aforementioned transfer mechanism is constituted not only to possess a selection mechanism which selects, by preliminarily investigating the load states of the aforementioned multiple information processing machines, information processing machines with which the rasterizing routine can be shared but also to transfer the aforementioned partial files into the aforementioned selected information processing machines.

(19)日本国特許庁（ＪＰ）　(12)特　許　公　報（Ｂ２）　(11)特許番号
特許第3168580号
(P3168580)

(45)発行日　平成13年５月21日(2001.5.21)　(24)登録日　平成13年３月16日(2001.3.16)

| (51)Int.Cl.⁷ | 識別記号 | ＦＩ | |
|---|---|---|---|
| Ｇ０６Ｆ　3/12 | | Ｇ０６Ｆ　3/12 | Ｄ |
| Ｂ４１Ｊ　2/485 | | Ｂ４１Ｊ　5/30 | |
| 5/30 | | Ｇ０６Ｆ　15/72 | |
| Ｇ０６Ｔ　11/00 | | Ｂ４１Ｊ　3/12 | |

請求項の数39(全 50 頁)

(21)出願番号　特願平3－507018

(86)(22)出願日　平成３年４月５日(1991.4.5)

(86)国際出願番号　ＰＣＴ／ＪＰ９１／００４５６
(87)国際公開番号　ＷＯ９１／１５８３１
(87)国際公開日　平成３年10月17日(1991.10.17)
審査請求日　平成10年２月13日(1998.2.13)
(31)優先権主張番号　特願平2－90728
(32)優先日　平成２年４月５日(1990.4.5)
(33)優先権主張国　日本（ＪＰ）

(73)特許権者　999999999
セイコーエプソン株式会社
東京都新宿区西新宿２丁目４番１号
(72)発明者　長坂　文夫
長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内
(74)代理人　999999999
弁理士　佐藤　一雄　(外３名)

審査官　田中　貞嗣

(58)調査した分野(Int.Cl.⁷，ＤＢ名)
G06F　3/12

(54)【発明の名称】　ページ記述言語インタープリタ



(57)【特許請求の範囲】
【請求項１】ソースファイルから印刷のための画素情報を得るラスタライズ処理機能を有する複数の情報処理機器を含む情報処理機器群を粗結合した情報処理システムであって、
前記情報処理機器群中のいずれかの情報処理機器が、
前記ソースファイル又はこのソースファイルから変換された中間コードファイルを、独立してラスタライズ処理することが可能な複数の部分ファイルに分割する分割手段と、
前記複数の部分ファイルうち、少なくとも１つを、ラスタライズ処理が可能な他の情報処理機器に転送する転送手段と、
を有する情報処理システム。
【請求項２】前記分割手段と転送手段を有する情報処理機器が、前記ラスタライズ処理機能をも有し、前記複数の部分ファイルのうち前記他の情報処理機器に送った残りの部分ファイルをラスタライズ処理して画素情報を得る請求項１記載の情報処理システム。
【請求項３】前記分割手段が、
前記ソースファイルに含まれる図形要素の相互間の重なりを検出する手段と、
前記検出された重なりを相互間に持つ図形要素が異なる前記部分ファイルに含まれないように分割を行う手段と、
を有する請求項１または２記載の情報処理システム。
【請求項４】前記転送手段が、前記複数の情報処理機器の負荷状態を予め調べてラスタライズ処理の分担が可能な情報処理機器を選択する選択手段を有し、前記選択された情報処理機器に前記部分ファイルを転送する請求項

1～3のいずれか一項に記載の情報処理システム。

【請求項5】さらに、前記情報処理機器群中のいずれかの情報処理機器が、前記他の情報処理機器のラスタライズ処理で得た部分的な画素情報を前記他の情報処理機器から集め、これらの集めた部分的な画素情報を組み合わせて全体的な画素情報を得る合成手段を有する請求項1～4のいずれか一項に記載の情報処理システム。

【請求項6】前記合成手段を有する情報処理機器が、前記ラスタライズ処理機能をも有し、前記合成手段は前記他の情報処理機器のラスタライズ処理で得た部分的な画素情報を前記ネットワークを通じて集め、これらの集めた部分的な画素情報と、前記部分ファイルのうち少なくとも1つを自身でラスタライズ処理して得た画素情報とを組み合わせて全体的な画素情報を得る請求項5記載の情報処理システム。

【請求項7】前記分割手段と転送手段を有する情報処理機器と、前記合成手段を有する情報処理機器とが、同一の機器である請求項5または6記載の情報処理システム。

【請求項8】前記分割手段と転送手段を有する情報処理機器がコンピュータとして構成され、前記合成手段を有する情報処理機器が印刷装置として構成された請求項5または6記載の情報処理システム。

【請求項9】前記情報処理機器群に印刷装置を含む請求項1～7のいずれか一項に記載の情報処理システム。

【請求項10】前記分割手段と前記転送手段が、前記分割手段と転送手段を有する情報処理機器のオペレーティングシステムの管理下で実行される1以上のプロセスとして実装される請求項1～9のいずれか一項に記載の情報処理システム。

【請求項11】前記ラスタライズ手段が、前記ラスタライズ手段を有する情報処理機器のオペレーティングシステムの管理下で実行される1以上のプロセスとして実装される請求項1～10のいずれか一項に記載の情報処理システム。

【請求項12】ソースファイルから印刷のための画素情報を得るラスタライズ処理機能を有する複数の情報処理機器を含む情報処理機器群を粗結合した情報処理システムであって、

前記情報処理機器群中のいずれかの情報処理機器が、

他の情報処理機器のラスタライズ処理で得た部分的な画素情報を前記他の情報処理機器から集め、これらの集めた部分的な画素情報を組み合わせて全体的な画素情報を得る合成手段を有する情報処理システム。

【請求項13】前記合成手段を有する情報処理機器が、前記ラスタライズ処理機能をも有し、前記合成手段が、前記他の情報処理機器のラスタライズ処理で得た部分的な画素情報を前記他の情報処理機器から集め、これらの集めた部分的な画素情報と、前記部分ファイルのうち少なくとも1つを自身でラスタライズ処理して得た画素情報とを組み合わせて全体的な画素情報を得る請求項12記載の情報処理システム。

【請求項14】前記情報処理機器群に印刷装置を含む請求項12または13記載の情報処理システム。

【請求項15】前記合成手段を有する情報処理機器が印刷装置として構成され、前記他の情報処理機器がコンピュータとして構成される請求項12～14のいずれか一項に記載の情報処理システム。

【請求項16】前記ラスタライズ手段が、前記ラスタライズ手段を有する情報処理機器のオペレーティングシステムの管理下で実行される1以上のプロセスとして実装される請求項12～15のいずれか一項に記載の情報処理システム。

【請求項17】他の情報処理機器と粗結合され得る情報処理機器において、

ソースファイル又はこのソースファイルから変換された中間コードファイルを、独立してラスタライズ処理することが可能な複数の部分ファイルに分割する分割手段と、

前記複数の部分ファイルのうちの少なくとも1つを、ラスタライズ処理が可能な他の情報処理機器に転送する転送手段と、

を有する情報処理機器。

【請求項18】前記分割手段が、

前記ソースファイルに含まれる図形要素の相互間の重なりを検出する手段と、

前記検出された重なりを相互間に持つ図形要素が異なる前記部分ファイルに含まれないように分割を行う手段と、

を有する請求項17記載の情報処理機器。

【請求項19】前記転送手段が、前記複数の情報処理機器の負荷状態を調べてラスタライズ処理の分担が可能な情報処理機器を選択する選択手段を有し、前記選択された情報処理機器に前記部分ファイルを転送する請求項17または18記載の情報処理機器。

【請求項20】前記他の情報処理機器におけるラスタライズ処理により得られた部分的画素情報を前記他の情報処理機器から受け、これら部分の画素情報を組み合わせて全体的画素情報を得る合成手段を、さらに有する請求項17～19のいずれか一項に記載の情報処理機器。

【請求項21】ラスタライズ処理して画素情報を得るラスタライズ処理機能をさらに有し、

前記複数の部分ファイルのうち、前記他の情報処理機器に送った残りの部分ファイルをラスタライズ処理して画素情報を得る請求項17～19のいずれか一項に記載の情報処理機器。

【請求項22】前記他の情報処理機器においてラスタライズ処理により得られた部分的画素情報を前記他の情報処理機器から受け、これら部分的画素情報と前記残りの部分の画素情報とを組み合わせて全体的画素情報を得る

5

合成手段を、
さらに有する請求項21記載の情報処理機器。
【請求項２３】コンピュータとして構成された請求項17～22のいずれか一項に記載の情報処理機器。
【請求項２４】印刷装置として構成された請求項17～22のいずれか一項に記載の情報処理機器。
【請求項２５】前記分割手段と前記転送手段が、オペレーティングシステムの管理下で実行される１以上のプロセスとして実装される請求項17～24のいずれか一項に記載の情報処理機器。
【請求項２６】前記ラスタライズ手段が、オペレーティングシステムの管理下で実行される１以上のプロセスとして実装される請求項21または22記載の情報処理機器。
【請求項２７】他の情報処理機器と粗結合され得る情報処理機器において、
前記他の情報処理機器においてラスタライズ処理して得られた複数の部分的画素情報を前記他の情報処理機器から受け、これらの画素情報を組み合わせて全体的画素情報を得る合成手段を有する情報処理機器。
【請求項２８】ソースファイル又はこのソースファイルから変換された中間コードファイルから、印刷のための画素情報を得るラスタライズ処理機能をさらに有し、
前記合成手段が、前記他の情報処理機器において得られた少なくとも１つの部分的画素情報を前記他の情報処理機器から受け、この部分的画素情報と自身でラスタライズ処理して得た画素情報とを組み合わせて全体的画素情報を得る請求項27記載の情報処理機器。
【請求項２９】コンピュータとして構成された請求項27または28記載の情報処理機器。
【請求項３０】印刷装置として構成された請求項27または28記載の情報処理機器。
【請求項３１】前記ラスタライズ手段が、オペレーティングシステムの管理下で実行される１以上のプロセスとして実装される請求項28記載の情報処理機器。
【請求項３２】他の情報処理機器と粗結合され得る情報処理機器において、
前記他の情報処理機器から送られる翻訳処理への参加を要求するメッセージに応答して、負荷状態に応じて前記参加が可能か否かを判断し、判断結果を示す情報を前記他の情報処理機器へ返信する手段と、
前記参加が可能なとき、前記他の情報処理機器から送られるソースファイルの又はこのソースファイルから変換された中間コードファイルの部分ファイルを受けて、この部分ファイルをラスタライズ処理して部分的な画像情報を得る手段と、
を有する情報処理機器。
【請求項３３】さらに、前記部分的な画素情報を前記他の情報処理機器へ送る手段を有する請求項32記載の情報処理機器。
【請求項３４】ソースファイル又はこのソースファイル

6

から変換された中間コードファイルを印刷のための画素情報に変換するラスタライズ処理機能を有し、他の情報処理機器と粗結合され得る情報処理機器であって、
自機器で印刷要求の発生した第１のソースファイル又はこの第１のソースファイルから変換された第１の中間コードファイルのラスタライズ処理を部分的に前記他の機器に担当させるためのクライアント手段と、
前記他の機器で印刷要求の発生した第２のソースファイル又はこの第２のソースファイルから変換された第２の中間コードファイルのラスタライズ処理を部分的に自機器で担当するためのサーバ手段と、
与えられたファイルについてのラスタライズ処理を実行するラスタライズ手段とを有し、
前記クライアント手段は、
前記第１のソースファイル又は前記第１の中歓呼コードファイルを、独立してラスタライズ処理することが可能な複数の部分ファイルに分割する手段と、
前記複数の部分ファイルのラスタライズ処理を部分的に前記他の機器に担当させるさせるために、前記複数の部分ファイルのうちの一部分を前記他の機器に送る手段と、
前記複数の部分ファイルのうち前記他の機器に送った部分を除く残りの部分を前記ラスタライズ手段に送る手段と、
前記他の機器でのラスタライズにより得られた部分的な画素情報を前記他の機器から受け、これと前記ラスタライズ手段から得られる前記残りの部分についての画素情報とを組み合わせて全体的な画像情報を得る手段と、を有し、
前記サーバ手段は、
前記第２のソースファイルの又は前記第２の中間コードファイルの部分ファイルを、前記他の機器から受けて前記ラスタライズ手段に送る手段と、
前記ラスタライズ手段により得られた前記部分ファイルについての画素情報を前記他の情報処理機器に送る手段と、
を有する情報処理機器。
【請求項３５】粗結合された複数台の情報処理機器を利用してソースファイルから印刷のための画素情報を得る情報処理方法において、
前記ソースファイル又はこのソースファイルから変換された中間コードファイルを、独立してラスタライズ処理可能な複数の部分ファイルに分割する過程と、
前記複数の部分ファイルを前記複数の情報処理機器で分散的にラスタライズして複数の部分的画素情報を得る過程と、
前記複数の情報処理機器で得られた前記複数の部分的画素情報を集め、それら集めた部分的画素情報を組み合わせて全体的な画素情報を得る過程と、
を有する情報処理方法。

7

【請求項３６】前記複数の情報処理機器にコンピュータ及び印刷装置を含む請求項35記載の情報処理機器。

【請求項３７】他の情報処理機器と粗結合された情報処理機器において、ソースファイルから印刷のための画素情報を得る情報処理方法において、

前記ソースファイル又はこのソースファイルから変換された中間コードファイルを、独立してラスタライズ処理可能な複数の部分ファイルに分割する第１の通過と、

前記複数の部分ファイルのうちの少なくとも１つを、前記他の情報処理機器にラスタライズさせるために前記他の情報処理機器へ送る第２の過程と、

前記複数の部分ファイルのうち前記他の情報処理機器へ送った残りの部分ファイルを、自身でラスタライズする第３の過程と、

（１）　前記他の情報機器でのラスタライズで得た部分的な画素情報を前記他の情報処理機器から集め、それら集めた部分的な画素情報と、自身でラスタライズ処理して得た画素情報とを組み合わせて全体的な画素情報を得る、

（２）　前記他の情報処理機器に前記全体的な画素情報を得させるために、自身でラスタライズ処理して得た画素情報を前記他の情報処理機器に送る、

の（１）又は（２）のいずれか一方を行う第４の過程とを有する情報処理方法。

【請求項３８】前記他の情報処理機器がコンピュータ又は印刷装置である請求項37記載の情報処理方法。

【請求項３９】前記第１、第２、第３および第４の過程が、前記情報処理機器のオペレーティングシステムの管理下で実行される１以上のプロセスとして実装される請求項37または38記載の情報処理方法。

【発明の詳細な説明】

技術分野

本発明は、コンピュータ周辺装置として用いられる印刷装置の、印刷制御を記述した記述言語プログラムの翻訳処理に関する。より詳しくは、本発明は、印刷制御手順、文字図形、画像情報などを記述し、それらの印刷画素への変換と印刷を一般化する目的で開発された記述言語（一般に「ページ記述言語」と呼ばれ、この明細書でも「ページ記述言語」と呼ぶ）の翻訳処理を行うインタプリタに関し、ネットワークによって複数の情報機器を接続したシステムに適用されるものである。

背景技術

電子写真方式、インクジェット方式などの高速デジタル印刷に適した印刷装置の開発に伴い、従来の文字情報中心の印刷から脱却した、画像、図形、文字などを同様に取り扱い、文字図形の拡大、回転、変形などが自由に制御できる１ページ単位での印刷手順制御方式が一般に普及してきた。このような制御方式の実現においては、制御手順を記述するプログラム言語を開発し、印刷装置あるいはコンピュータにその言語の翻訳処理系（いわゆるインタープリタ）を搭載し、このインタプリタによりその言語から印刷画素への変換を行なうのが一般的である。

8

１台のコンピュータにおいてアプリケーションプログラムが発生した印刷要求は、オペレーティングシステムの機能を呼び出して印刷装置に働きかけようとする。このとき印刷装置の解像度、印刷方式、印刷ヘッドのドット数などのハードウェアに依存せずに印刷手順を行なうために使用するのが「ページ記述言語」である。

従来のシステムの一例を示す図1A、1Bを参照して説明すれば、コンピュータ6d内でアプリケーションプログラムが発生するページ記述言語のソースコード19は、ページ記述言語インタープリタ８によって翻訳され実際の印刷に使用される画素に置き換えられる。このとき、ページ記述言語インタープリタ８が図2Aのようにコンピュータ6e内で実行されるか、或いは図2Bのように印刷装置21内で実行されるかは、システムによって異なる。しかし、いずれにしても、単一のプロセッサによりページ記述言語インタープリタが実現されるのが従来のシステムである。

また本発明に関連する従来例としてプリントサーバと呼ばれるシステムがある。このシステムは、例えば図1Aにおいて、コンピュータ6dにさらに図示しない他のコンピュータとネットワークで接続されるているもので、プリント処理要求の発生したコンピュータ6dに代わり、他のコンピュータがページ記述言語のラスタライズを行なうものである。この場合にも、他のコンピュータのプロセッサが単独でインタプリタを実現している。

インタプリタが行うページ記述言語から印刷画素への変換には、大容量のメモリが必要である。また、ソフトウェアのみでインタプリタを実現する場合、処理時間が長くかかることも重要な問題である。加えて、今後カラーページプリンタの普及が予想され、白黒印刷にあっても高密度な印刷への要求が高まることが明白である。これらはどちらもより大きな記憶空間を必要とする。さらに、カラー印刷においては印刷制御手順はより複雑になり、プログラムの翻訳処理系に要求される事項もその実行により長時間が必要な処理となるであろう。

本発明はこれら問題点の解決を目指したものであり、インタプリタの各部分の実行を複数のプロセッサによって分散処理することでプロセッサ１個あたりの処理すべき内容を軽減すると共に、それら複数のプロセッサが各々担当するメモリ空間を利用することにより全体として大きなメモリ空間を確保するのに等しい効果をあげ、もって高速高密度処理を実現することを目的とする。

発明の開示

本発明はソースファイルから印刷のための画素情報を得るラスタライズ処理機能を有する複数の情報処理機器を含む情報処理機器群を粗結合した情報処理システムであって、

前記情報処理機器群中のいずれかの情報処理機器が、

前記ソースファイル又はこのソースファイルから変換された中間コードファイルを、独立してラスタライズ処理することが可能な複数の部分ファイルに分割する分割手段と、

前記複数の部分ファイルうち、少なくとも１つを、ラスタライズ処理が可能な他の情報処理機器に転送する転送手段と、

を有する情報処理システムを提供する。

図面の簡単な説明

図1A及び図1Bは従来のPDLインタープリタの構成を示すブロック線図。

図２は本発明のPDLインタープリタの第１の実施例の構成を示すブロック線図。

図３は図２の実施例におけるコンピュータの構成を示すブロック線図。

図４は図３におけるクライアントプロセスの構成を示すブロック線図。

図５は図３におけるサーバプロセスの構成を示すブロック線図。

図６は図４のクライアントプロセスの動作を示す流れ図。

図７は図３における常駐プログラムの動作を示す流れ図。

図８は常駐プログラムの構成を示すブロック線図。

図９は応答メッセージの仕様の説明図。

図10はプリンタ構造体を含む応答メッセージの仕様の説明図。

図11は多角形描画手順の説明図。

図12は多角形の塗りつぶし例の説明図。

図13は塗りつぶしアルゴリズムの流れ図。

図14及び図15は塗りつぶしアルゴリズムの説明図。

図16は図形グループ判別の説明図。

図17及び図18は塗りつぶし例の説明図。

図19及び図20は図形グループ判別方法の第１の例の説明図。

図21は図19及び図20の例における図形要素の重なり検出手順の流れ図。

図22は図19及び図20の例における図形グループの判別／領域結合検出の流れ図。

図23及び図24は図22の図形グループ間の領域結合検出手順の説明図。

図25は図形グループ検出方法の第２の例の説明図。

図26は図25の方法における検出された図形グループの説明図。

図27は図形グループ検出方法の第３の例の説明図。

図28は印刷手順の説明図。

図29は印刷手順の流れ図。

図30は翻訳処理手順の流れ図。

図31は本発明の第２の実施例の構成を示すブロック線図。

図32及び図33は図30の第２の実施例の動作を示す流れ図。

図34は本発明の第３の実施例の構成を示すブロック線図。

図35は本発明の第４の実施例の概略構成を示すブロック線図。

図36は本発明の第５の実施例の概略構成を示すブロック線図。

図37は本発明の第６の実施例の構成を示すブロック線図。

図38は図37の第６の実施例におけるプリンタの構成を示すブロック線図。

発明を実施するための最良の形態

最近のコンピュータシステムにおいては、ネットワークを用いて複数台のパーソナルコンピュータやエンジニアリングワースステーションなどを接続し使用するのが一般的に成りつつある。本発明は、このような環境においてネットワーク上のどれか１台のコンピュータで実行されているアプリケーションプログラムが印刷処理に関する要求をオペレーティングシステムに対して発生したとき実行されるものである。どれか１台のコンピュータにおいて発生した印刷要求は、オペレーティングシステムの機能を呼び出して印刷装置に働きかけようとするが、このとき印刷装置の解像度、印刷方式、印刷ヘッドのドット数などのハードウェアに依存せずに印刷手順を行なうために「ページ記述言語」を使用する。言い換えれば、アプリケーションプログラムは、ネットワークシステムで使用されている印刷装置のハードウェアの内容を知らないため、ページ記述言語によって書かれたソースファイルを翻訳プログラムに渡す。本発明においては、オペレーティングシステムの管理下で翻訳プログラムが前記ソースファイルを分解可能な小ファイルに分割し、各小ファイルの処理をネットワーク上で利用できる他のプロセッサに分担させて印刷画素への変換を行なわせる。

以下に本発明の幾つかの好適な実施例を説明する。

図２に本発明のページ記述言語並列処理インタープリタの第１の実施例の構成を示す。本実施例は、複数台のコンピュータ6a～6cと、少なくとも１台のプリンタ21により構成される。図３は、一つのコンピュータ6aを取り上げて、それが持つ機能を図３に示す。他のコンピュータ6b, 6cも同様の機能を持つ。

この実施例において、本発明のページ記述言語並列処理インタープリタを動作させるハードウェアの構成を一般的に述べれば以下の通りである。

（ア）印字要求の生じるコンピュータ（以下「コンピュータＡ」という）とは別個に、少なくとも１台以上の情報処理機器（例えば他のコンピュータや印刷装置）がコンピュータＡとなんらかの通信手段によって接続されて

いる。

（イ）コンピュータＡおよびそれに接続される情報処理機器が本発明の分散処理ページ記述言語インタープリタを実行可能である。

（ウ）コンピュータＡおよび情報処理機器が少なくとも２つ以上のプロセスを実行できる。

ここでオペレーティングシステムの管理下において実行に必要なメモリ、プロセッサ資源を与えられ動作するプログラムの実行単位を「プロセス」と呼ぶ。プロセスは、メモリ割り当て及び、プロセスヘッドにより管理される。プロセスヘッダには、プロセスの識別子（プロセスID）現在のプログラムカウンタ、スタックポインタ、その他CPUレジスタの保存値及び、メモリ状態が記録される。実行順位の低いプロセスは、主記憶装置に置かれず補助記憶装置に退避されるが、これらの管理はオペレーティングシステムに依存する。また単一のプログラムが、複数個のプロセスから成立する構成も認められる。また以下の説明においては印字要求の発生するところのコンピュータＡと何らかの通信手段によって接続される他の情報機器との間の通信手段および該通信手段を実現するハードウェアを「ネットワーク」を呼称する。ネットワークは、ハードウェアに依存した物理的なアドレスと、上層のプログラムから参照される論理的なアドレス管理機構をもつ。これらアドレスは、ネットワーク上の特定のコンピュータへのパケット転送に用いられるが、多くのネットワークプロトコルにおいて、より抽象化の進んだ上層の処理手順を持っている。本発明の実施例においては、これら上層の処理ルーチン群が、オペレーティングシステムによって供給されているものとして、説明を行なう。この様な管理機構により、プログラムは、コンピュータを名前と論理アドレスにより検出可能であり、また通信相手のプロセスとは、通信ソケットもしくは類似の機構を用いたパケトの送信／受信が可能であるものとする。

次に図３を参照して、この実施例におけるコンピュータ6aの構成を説明する。他のコンピュータ6b, 6cの同様の構成である。

このコンピュータ6aは、ページ記述言語の翻訳処理系の一部を構成する。ここで言うところの「ページ記述言語」とは、ドットマトリクス方式、電子写真方式などによる印刷装置において、印字データの配置、形状、文字情報などの制御手順を記述するために使用される言語である。言語の文法は一般に各言語についてその独自の仕様があり、約束された取り決めに過ぎない。本発明は特定のページ記述言語により記述された書類を翻訳し実印刷データに置き換える処理方式についてなされたものであるから言語仕様については特に言及しない。以下「ページ記述言語」をPDLと略す。

本発明の翻訳処理系は図３にて参照番号１と２によって示される２つの部分を含む。１は小規模な常駐プログラムであり、２が印刷データ生成の処理を行なう部分、すなわちPDL並列処理インタープリタである。この実施例においては印刷装置21（図２参照）は印刷に際し、画素の印刷可否のみをデータ列中の１ビットの0/1によって指定するいわゆるビットイメージ印刷のみを行なうものである。これに合わせて、PDL並列処理インタープリタ２は、ページ記述言語のソースコードを入力とし、ビットイメージ印刷のためのデータ列およびプリンタ制御コードを出力する。



本実施例のPDL並列処理インタープリタ２は、３つのプロセス210, 211, 212を含む。プロセス210は、PDL翻訳のためのクライアントプロセスであり、ラスタライザロセス212を使用する。またプロセス211は、サーバプロセスであり、常駐処理プログラム１により生成される。プロセスの生成はオペレーティングシステム４のプロセス生成機構207で行なわれる。

オペレーティングシステム４は、スケジューラ208により、複数のプロセスに対するCPU時間の割り当てを行なう。このため、コンピュータ6aは、複数のプロセスを同時に実行可能であり、いわゆるマルチタスク処理を実現している。この様なスケジューラ208は、オペレーティングシステム４のカーネル206のプライオリティ管理、待ち行列管理処理により実現される。またオペレーティングシステム４では、ハードディスク装置ドライバ、クロックドライバ等のデバイスドライバ３が使用される。常駐プログラム１およびPDL並列処理インタープリタ２はネットワークデバイスドライバ213を使用し、このネットワークデバイスドライバ213は他のコンピュータ6b, 6cや印装置21とデータ列の交信を行なう。印刷処理に使用される他の資源として、字体情報205がある。この字体情報205は、アウトラインフォントデータ又はビットマップフォントデータの形で、補助記憶装置204に記録されている。

コンピュータ6aにおいて、複数個のアプリケーションプログラム５、18が同時に実行可能である。コンピュータ使用者は、マウス、キーボード等の入力装置23を使用し、アプリケーションプログラム５、18に対する操作を行なう。ユーザインターフェースプロセス201は、使用者の操作を検出し、ユーザイベント待ち行列202へ、ユーザ事象をエンキューする。一般にフォアグランドで実行されるアプリケーションは、イベント処理プロセス203から現在イベントを取得するプログラム（例えば、アプリケーションプログラム18）である。使用者が、アプリケーションプログラム18に対して行なう印刷処理要求は、イベント処理プロセス203、あるいはコマンドインタープリタ等により検出され、アプリケーションプログラム18に伝達される。本実施例においては、この結果、アプリケーションプログラム18が、PDLソースコードにより記述された出力19を生成する。PDL並列処理インタープリタ２は、この出力を受け取る。

使用者が入力装置23を用いて、アプリケーションプログラム18に対し、印刷処理を要求すると、PDLにより記述されたファイル19が出力される。ファイル19は、コンピュータ6aのファイルシステム管理下で、特定のディレクトリに置かれ、一定の時間間隔で、このディレクトリ内のファイルを検出する機構により取り出され、PDL並列処理インタープリタ２のプロセス210に渡される。使用者の立場から見た印刷処理は、これにより完了である。プリンタ21からの印刷出力12を得ることで、使用者の印刷要求が実現される。この時、本実施例の構成においては、印刷要求が発生したコンピュータ6aが、そのPDL翻訳処理をこのコンピュータ6aと他のコンピュータ6b、6cに分散し、画素生成をそれら複数台のコンピュータでの並列処理により行なう様に動作する。

この動作を図３、図４及び図５により説明する。図４は、PDL並列処理インタープリタ２の内部での、プロセス210および212の機能の説明図である。また図５は、PDL並列処理インタープリタ２の内部での、プロセス211および212の機能の説明図である。

図３において、３台のコンピュータ6a、6b、6cがネットワーク７により結合されている。使用者による印刷要求操作はコンピュータ6aを用いてなされたものとする。コンピュータ6aの並列処理インタープリタ２の内部では、図４を参照して、まず一次翻訳処理215が、PDLで記述されたソースファイル19からコンピュータの処理により適した中間コード形式のファイル10を生成する。PDLの仕様の中には、実際のプリントエンジンの画素の解像度に依存しない記述を実現するため、図形記述の際にユーザ座標系と呼ばれる任意の座標系の指定を認めるものがある。一方、実際の印刷画素は、プリントエンジンの物理的な仕様に従った、デバイス座標系上の画素として扱われるため、これらの解像度の変換を行ない、実画素データの座標系に合わせた記述を得る必要がある。またPDLの述語は、一般に文字列形式であるため、冗長度が高い。一次翻訳処理215は、座標系及び述語をより処理し易いデータ形式に変換する。

生成された中間コード10は、グループ検出処理216を経て複数の部分に分割され、各部分は互いに独立した中間コードであるかのように扱われる。後述する様に、この分割は、中間コード10をできるかぎり多くの小部分に分割する方針で行われる。この分割は印字要求の発生したコンピュータ6aで、クライアントプロセス210のグループ検出手段216により行なわれる。分割された中間コード列230の各部分は、グループ送信ソケット222を介し、ネットワーク７に出力され、他のコンピュータ6b、6cの各々の分散処理インタープリタ２に転送される。実際にこれらデータ列を受け取るのは、図５にその構成を示したサーバプロセス211である。

図５を参照して、他のコンピュータ6b、6cの各サーバプロセス211は、グループ受信ソケット225を介し、前記



中間コードデータ列を受信し、グループ受信処理227でデータ部分を抜き出し、ラスタライザ212に転送する。ラスタライザ212は実画素生成のプロセスで、曲線または直線の記述から、点列を発生し、実印刷データに相当するデータをメモリ上に発生する。このデータは、画素データ転送処理228で圧縮され、画素データ送信ソケット226から、ネットワーク７に出力される。

再び図４を参照して、一方、コンピュータ6aのクライアントプロセス210では、他のコンピュータ6b、6cの各サーバプロセス211からの画素データを、画素データ受信ソケット223を用い受信し、イメージメモリ領域229に蓄積する。また前記分割された中間コード列230の一部分は、直接ラスタライザ212に転送し、画素データに変換する。これら画素データが、画素合成処理220により合成され、印刷処理データが完成する。

上記一連の処理が完了した後、クライアントプロセス210は、画素データを、プリンタ21の物理的仕様に合わせ、プリンタ制御コードの付加処理221を行ない、印刷データ信号ソケット224から、ネットワーク７を介しプリンタ21へ送信する。

図２を参照して、プリンタ21のネットワークデバイスドライバ303は、ネットワーク７を通し受信したデータ列を、エンジンコントローラ302へ伝達する。エンジンコントローラ302は、画素データのビット配置などを、プリントエンジン301の物理的仕様に合うよう置き換え、印刷画素データをプリントエンジン301に出力する。この結果、印刷出力12が得られことになる。

以上のような処理手順を実現するために、本発明のPDL並列処理インタープリタは２つの特徴的な部分を備える。これを、図３と図６および図７の流れ図を参照して説明する。

アプリケーションプログラム18の発した印刷要求がオペレーティングシステム４を通じて並列処理インタープリタ２に伝達されると、並列処理インタープリタ２はできるかぎり多くのプロセッサーを使用して印刷手順を行なおうとする。使用可能なプロセッサーは他のコンピュータ6b、6cの内部にあるであろうし、さらに、印刷装置21内のプロセッサも利用できるかもしれない。分散処理インタープリタ２にとって、それら他の情報処理機器内のプロセッサが利用できるか否かの区別は、それら他の情報処理機器に対して転送した特定のフォーマットのメッセージに対してそれら他の情報処理機器が応答するか否かで決められる。

即ち、印刷要求の発生したコンピュータ6aでは、並列処理インタープリタ２内のクライアントプロセス210が、ネットワーク上のコンピュータ6b, 6cのアドレスをアドレス管理テーブル214から取得し、分散処理に参加できるコンピュータを募る（ステップ13～16）。この場合、クライアントプロセス210は、他のコンピュータ6b, 6cに対して、分散処理に参加することを要求するメッセ

ージをネットワークを通じて送る。他のコンピュータ6b, 6cでは、ネットワーク７を通して分散処理に参加することを要求するメッセージが届いた場合、常駐プログラム１が動作してそのコンピュータ6b, 6c中での現在のプロセッサーの負荷を判断し、そして、必要なだけのメモリ領域の確保とプロセスの生成に障害が無ければ、分散処理に参加可能である旨を返信する（ステップ25～29）。

この返信を受けて、印刷要求が生じたコンピュータ6aはネットワーク上で分散処理に参加可能な情報処理機器（この実施例ではコンピュータだけであるが、他の実施例ではプリンタも利用できる場合がある）だけを利用しそれ以降の手順を進めるように動作する。ここで分散処理に利用可能であった情報処理機器のネットワーク上でのアドレスは、図４に示したサーバマシン管理表219に登録される。

このようなネットワークを用いた処理においては、複数の情報処理機器から同時に印刷に関しての要求、あるいはその他の要求が発生するのは充分に考えられることである。この様な要求の競合状態においては、本発明の分散処理インタプリタを有する全ての情報処理機器が直ちに要求に応じられる訳ではない。図８は常駐処理プログラム１の構成図である。プロセッサの負荷は、オペレーティングシステム４のスケジューラ208に対する要求の待ち行列209の長さを検出することで判断できる。常駐プログラム１の負荷検出機構324は、プロセス間通信により、待ち行列209の長さを知ってプロセッサの負荷の程度を判断し、クライアントプロセス210からの要求に応答を返す動作をする。

以下に、この動作を図６および７の流れ図と図８とを参照して説明する。

前述のようにネットワークの上でどのような情報処理機器が結合されているかは、それぞれ情報機器についてネットワーク上で固有に与えられた機器ごとの名称によって区別可能であり、各コンピュータはネットワークプロトコルにおいて用意されるブロードキャスト機能などを使用して各機器の名称を知り、これを名前表ネットワーク管理テーブル214（図４参照）に格納しているものとする。そこで印刷要求の発生したコンピュータは、図６のステップ13に示すように、ネットワーク上のどれか１つの情報処理機器に対し、サーバプロセス211を起動させることが可能か問い合わせるメッセージを送る。

その情報処理機器内では、図８に示すように、ネットワークデバイスドライバ213を介して受け取られたメッセージは、デバイスドライバインターフェース321で取得され、要求メッセージ解析処理322に入る。このメッセージを受け取った情報常駐処理プログラム１は、図７の流れ図に示されるように動作する。すなわち、まずステップ25で、メッセージの種類が状態の確認と返信を要求するものであるかどうか判断し、Yesである場合はステップ26にてプロセッサの使用頻度を調査し、その負荷が新たなプロセスを生成するのに問題ないかどうかステップ27で判断する。これが問題ない場合は、続いてステップ28でメモリの使用状況を調べ、新たなプロセスを作れるかどうかをステップ29で確認する。全てに問題なければ正常な終了コードが返送され、ステップ27またはステップ29の判断において何らかの障害があればエラーコードが返送される。

正常な終了コードが返されたときは、印刷要求の発生したコンピュータでは、受諾のメッセージを返信されたものと見なされる。また本発明の並列処理インタープリタが組み込まれていない情報処理機器からは受諾メッセージが戻って来ない。この判断は図６のステップ14においてなされる。ここで受諾メッセージが返信された場合は、ステップ15に示すように前記の応答をした情報処理機器に対し再びメッセージを送り、プロセスを起動し分散処理インタープリタを動作させるように促す。このメッセージを受け取った情報処理機器は、図７のステップ30においてプロセス生成要求のメッセージかどうか判断し、Yesの場合はステップ31で新たなプロセス211を生成する。図８の構成では、起動処理323が実施され、オペレーティングシステム４のプロセス生成機構207により、サーバプロセス211が生成される。

印刷要求の発生したコンピュータのクライアントプロセス210は、ステップ16にてネットワーク上の全ての情報処理機器に対し問い合わせたかネットワーク管理テーブル214により判断し、未問い合わせの情報処理機器がに対しては、ステップ13、14及び15の処理を繰り返す。全ての機器についての問い合わせが完了した場合には、自らが分担すべきPDLの翻訳処理（ステップ17）を開始する。この処理は前述の様に、クライアントプロセス210が、ラスタライザ212を使用し行なうものである。

次に本実施例のPDLの分散処理インタープリタを実行する際に要求されるソフトウェアの動作の詳細について記す。以下の説明は主として次の４点について行なわれる。

（A）印刷要求の発生したコンピュータがネットワーク上の翻訳分散処理を行なう各コンピュータの補助記憶装置にいかなる種類のフォントが記録されているかを知ること。

（B）ネットワーク上において現在実行中の「印刷要求」が発生するPDLソースファイルの翻訳分散処理を行なう全てのコンピュータに対して、印刷の実際の動作が行なわれる印刷装置の、解像度、用紙サイズなどの物理的条件を共通に知らせること。

（C）PDLが発生したソースファイルを分割して分割した各部分を独立に画素へ翻訳し、それらを後の別手順において重ね合わせても、全部分を逐次画素に置き換えた４場合とまったく同じ印刷画像が得られよう、ソースファイルから図形グループを検出してグループ毎に分割を

行なうこと。

（D）ソースファイルの分割した各部分の翻訳処理を、ネットワーク上のどの情報処理機器に分担させたかを記録し、かつその実行状況を把握すること。

図２を参照して、印刷要求の発生したコンピュータ6aが、ネットワーク上の複数の他のコンピュータ6b、6cに対して文字コードを転送し、そしてそれらのコンピュータ6b、6cから翻訳処理により得られたビットマップを受け取る、というプロセスを実行するとき、転送される文字コードには、実際の文字列の他に文字種、大きさおよび文字飾りに関する情報が付加されている。例えば、「関東地方の天候は、」という文字列を印刷データに変換するためのに転送される情報は、｛文字種＝細明朝体｝＋｛ポイント数＝12｝＋「関東地方の」＋｛文字種＝細明朝体｝＋｛ポイント数＝12｝＋｛文字飾り＝斜体｝＋「天候は、」という形式をとることができる。このような形式の文字情報が今コンピュータ6cに転送されたとしても、コンピュータ6cに、細明朝体の印刷文字データを作り出すための字体データが存在しなければビットマップは生成できない。ここで言うところの「字体データ」とは、外形形状を規定し、外形を表す曲線形式を取り決めた元データから実際の印刷に使用する画素を生成するいわゆる「アウトラインフォント」であっても、或いは予め定まった大きさについて各文字を表すために与えられた画素の0/1データの集まりである「ビットマップフォント」であっても構わない。少なくともコンピュータ6aはコンピュータ6cに｛文字種＝細明朝体｝、｛ポイント数＝12｝であるような文字を処理する手段が存在することを知った上で、上記文字列の印刷データへの変換処理を6cに要求するものでなければならない。一つの書類には複数個の文字種が使用され、それら全ての文字種が全てのコンピュータで利用可能であるかどうかは印刷処理の直前まで不明である。なぜならコンピュータの使用者が自分で不要となった文字種のデータを廃棄することは、いつの時点でも可能な処理だからである。本実施例はかかる点で不都合が生じないようにメッセージの受渡しにより次のような処理を行なう。

現在発生している印刷要求のための分散処理に応ずることのできるネットワーク上の情報処理機器は、図７の流れ図に示したステップ31までの全処理を正常に実行したものである。これら機器のネットワークアドレスは、既に図４に示すサーバマシン管理表219に格納済みである。ここまでの図７の処理は、サーバ側の情報処理機器（例えばコンピュータ6c）において、図３に示した常駐処理プログラム１によって行なわれる。この後、実際にネットワークから転送されてくるメッセージに従い、PDL並列処理インタープリタ２のラスタライザプロセス212を制御して、印刷データの生成を実際に行なうのは、図７のステップ31で作られたサーバプロセス211である。サーバプロセス211は性質上アプリケーションプログラムの１つの様に振る舞うことになる。すなわちオペレーティングシステムのシステムコールを利用し自分に必要なメモリ、入出力チャンネルなどを確保する。このサーバプロセス211は、ユーザインターフェースとのインタラクティブな動作を伴わないため、使用者の気づかないうちに開始され終了するという点が、多くの一般的なアプリケーションプログラムと異なる点である。

図５に戻り説明を続けると、クライアント側コンピュータのクライアントプロセス210はサーバ側コンピュータのサーバプロセス211に対して以下の４種類のメッセージを発行し、これらメッセージがグループ受信ソケット225によりネットワークから受信される。サーバプロセス211は要求メッセージ解析処理231により、受信したパケットからメッセージを取り出し、それらに応答する。応答可能な上記４種類のメッセージを示す。

字体要求
処理要求
状態要求
取得要求

これらメッセージにつき順次説明する。

（１）　字体要求

プロセス211は字体要求のメッセージを受け取ると字体要求処理232において、字体情報205にアクセスし、直ちに現在所持している全てのフォントの字体データファイルを「変更不可、廃棄不可」の状態にし、図９に示す仕様の応答メッセージを返送する。メッセージは次の４つの部分からなっている。まずヘッダがあり、これはプロセス210に対する応答であることを示し、続いて応答メッセージが含む構造体数、該構造体数によって示された個数だけの構造体、最後に終了符号、という構成である。ここで応答として返される構造体は、本実施例においては、表１に示した構造体メンバからなるデータ列である。

表１

| 構造体メンバ | 変数型 | 変数の意味 |
|---|---|---|
| FontName | string | フォントの名称 |
| BOSelection | integer | =0 のときビットマップフォント |
| Size | integer | サイズ（本文注） |
| FDPtr | longint | フォントデータの格納されている番地 |
| CurvAlgo | word | 補間曲線処理のトラップワード |
| FixAlgo | word | 終端処理のトラップワード |
| FillAlgo | word | 塗りつぶし処理のトラップワード |
| ProcPtr | longint | 追加処理の手続き開始番地 |

表１中の変数型stringは文字列を表し、変数型integerは16ビットの整数、wordは16ビットの符号なし整数、longintは32ビットの整数であり、構造体メンバBOselectionは＝０の場合ビットマップフォントを使用することを示し、構造体メンバSizeはビットマップフォントの場合はその大きさを示し、アウトラインフォントの場合はヒンティング処理を行なうべき上限の大きさを示している。

以上の様な応答メッセージを受信したクライアントプロセス210は、ネットワーク上で分散処理に参加可能な情報処理機器群が、各々どのようなフォントを持っているか知ることが出来る。この結果、クライアントプロセス210では、フォント資源管理表218の更新が行なわれる。図４に示すように、プロセス210は、サーバマシン管理表219により、並列処理を行なうサーバコンピュータのネットワーク上のアドレス及び、通信ソケットの情報を管理する。上記の手順によりサーバコンピュータのアドレスに結び付けられたフォント資源の管理が行なわれるため、文字列処理要求をサーバプロセス211に送信する際には、該サープロセス211の動作するところの情報処理機器に存在するフォントセットについてのみ要求を発行するよう作用する。また一連の翻訳分散処理が完結した場合は、プロセス211にメッセージを送り、先にフォントデータに対し指定した「変更不可、廃棄不可」の属性を解除する様に働きかける。

（２）　処理要求

次にクライアントプロセス210は処理要求を各サーバプロセス211に発行する。プロセス211の処理要求に対する処理233は、グループ受信手段227に働きかけ、以下の処理を行なう。

処理要求は、具体的なPDLの個々の処理を要求するもので本実施例の場合、図10の仕様でメッセージが送信される。まずクライアントプロセス210からのメッセージであることを示すヘッダがあり、続いて「プリンタ構造体」と称する部分が続く。プリンタ構造体の構造体メンバは表２に示すものである。

表2

| 構造体メンバ | 変数型 | 変数の意味 |
|---|---|---|
| ClippingLTH | integer | 左上端のクリッピングの値（水平方向） |
| ClippingLTV | integer | 左上端のクリッピングの値（垂直方向） |
| ClippingRBH | integer | 右下端のクリッピングの値（水平方向） |
| ClippingRBV | integer | 右下端のクリッピングの値（垂直方向） |
| ResolutionH | integer | 水平方向の解像度 |
| ResolutionV | integer | 垂直方向の解像度 |
| RGB | integer | 色指定 |
| ProcPtr | longint | 追加処理の手続きの開始番地 |

すなわち長方形の印刷領域を印刷用紙上に設定し、該印刷領域の左上の座標と、右下の座標を４つの16ビット整数によって指定するためのメンバ（ClippingLTH、ClippingLTV、ClippingRBH、ClippingRBV）と、水平垂直方向の印刷密度を与える２つの16ビット整数（ResolutionH、ResolutionV）によって印刷装置の物理的仕様が伝達される。ここで印刷密度は、１インチあたりの画素数を表す値が用いられ、座標データは解像度に対する相対値である。例えば１インチあたり300画素の解像度の印刷装置であり、横方向に８インチまで印刷可能な印刷装置については、ClippingLTHとClippingRBHの値の差は2400である。プリンタ構造体の他のメンバの内、RGBは色処理のためのトラップワードとして使用される。ProcPtrはプリンタ固有の処理を追加するために手続きが必要となった場合の関数へのポインタを格納する場合の予約領域である。RGB及びProcPtrの各値の代入は、サーバ側のプロセス211の処理に任される。以上のようなプリンタ構造体に続いて実際のPDLのソースコードを処理した中間言語により構成された構造体が続き、終了符号が続く。なお上記説明中に使用した「トラップワード」とは、オペレーティングシステムの機能呼び出しに使用されるソフトウェア割り込み命令の命令語を意味する。但し本来プロセッサに用意された命令語によるソフトウェア割り込み以外にも、本実装命令、不正命令などが同様な目的に使用可能であることは周知の通りである。

（３）　状態要求／取得要求

クライアントプロセス210は以上のようにしてサーバプロセス211に対しPDLにより記述されたドキュメントの一部分の翻訳処理依頼を発生する。この後、クライアントプロセス210も該ドキュメントの別の一部分の翻訳を分担し処理する。クライアントプロセス210はこの処理が終了すると直ちに状態要求のメッセージを各々のサーバプロセス211に送信する。各サーバプロセス211では、状態要求のメッシージが処理234を起動し、画素データを生成状況がモニタされる。クライアントプロセス210は、処理終了のメッセージを返送してきた情報処理機器に対して取得要求メッセージを発生する。それら情報処理機器のサーバプロセス211では、取得要求メッセージが処理235を起動し、処理235は、画素データ送信処理228を制御して翻訳処理により生成された印刷データの転送を行なわせる。クライアントプロセス210は、それら情報処理機器から印刷データのビットマップを受け取

る。

ネットワーク上をサーバプロセス211からクライアントプロセス210に転送される印刷データは、ヘッダと終了符号により囲まれたビットマップのデータ列であり、データ列のバイト数、圧縮方法の指定はヘッダに含まれる。全てのサーバプロセス211から取得要求に応じた印刷データの返送があったことを確認し、クライアントプロセス210は、磁気ディスク装置などの補助記憶装置を用い、取得した印刷データの重ね合わせを行ない、その結果を印刷装置に転送し印刷出力を行なう。これら処理状況の管理には、グループ管理表217が使用される。これについては後述する。

次にPDLにより記述されたソースファイルを適切に複数部分に分割する手段について説明する。ここでは説明を明瞭にするためPDLの内の次の２つの「手続き」のみを使用した実施例を挙げて説明する。ここで「手続き」とはプログラミング言語の構文上の１つの実行単位であり、プログラミング言語Pascal（ANSI/IEEE770X3.97、ISO7185、英BS6192）において、通常「PROCEDURE」として扱われるものに同様である。他のプログラミング言語の類似の例としては、プログラミング言語Ｃの「戻り値の無い関数」、FORTRANの「サブルーチン呼び出し」などがこれに近いものである。

手続き:FillPolygon（引数　↑多角形）

手続き:InvertPolygon（引数　↑多角形）

上記手続きの使用に先だって、次のデータ型をあらかじめ定義する。

16ビットの整数　N、H、V

データ型　座標＝２要素からなる配列（H、V）

構造体　多角形＝（多角形の角数　N、N個の座標の集合）

構造体′多角形′を指し示すポインタ＝↑多角形

すなわち、N、H、Vは−32768、32767までの値をとる整数であり、１つの座標は水平方向の値H、垂直方向の値Vを要求する配列によって記述され、多角形とは整数Nで示されるN個の座標を持つ図形を記述するための構造体である。「多角形」は、前記のN個の座標を第１の座標から第Nの座標まで昇順に直線で結び、最後に再び、第Nの座標から第１の座標までを直線で結んで図形を閉じたものを言う。従ってN＝６であっても、いわゆる６角形とは限らない。また、前記多角形構造体が作りだす図形の内側を塗り潰す操作を行なった場合、第ｉ−１座標と第ｉ座標を直線で結ぶ操作を繰り返し行なって閉じた領域が生じたとしても、該閉領域が必ず塗りつぶされる部分とはならない。これは本実施例が用いた「塗りつぶしアルゴリズム」のためである。

この例を図11および図12を用いて示す。図11は、N＝６であるような多角形構造体から図形を書き表そうとした時の直線描画の手順を表した説明図である。多角形構造体には図11の□の中に書き表した順番に従って座標が保持されているものとする。従って図中の矢印の順番で直接を引き作りだされる多角形は６角形とはならない。またこの構造体によって書き表された多角形を塗りつぶした図形は図12で示される。黒色で表される領域が塗りつぶしの対象となった部分である。前述の様に直線で囲まれた閉じた領域であっても塗りつぶされていない部分が存在する。この理由を明らかにするため、本実施例において使用した「塗りつぶしアルゴリズム」について図13、図14、図15を用いて説明する。以下の説明では紙面に対して図12の矢印で示した方向を「水平」と見なすものとする。また簡単のため本実施例で行なう印刷処理は白黒二値画像のみとし、従って注目画素の印刷／非印刷の区別は１ビットの0/1の値で判断し、＝１である場合を印刷するものと定める。本発明で用いるPDLについては、紙面に対し水平な方向に８ビットからなる処理単位を設定し、該単位を「１バイト」と呼称する。実際の処理は32ビット（４バイト）を単位として行なう場合が多いが、図の説明では８ビット単位での扱いをする。

本実施例で使用した「塗りつぶしアルゴリズム」の流れ図を図13に示す。流れ図中で「水平座標の値がH、垂直座標の値がV」であるような位置に在る画素として扱われるべき１ビットの内容を（H、V）と書き表す。またN個の座標を順次数えるためにｉという変数を、水平軸上の位置を与えるためにｘ、垂直軸上の位置としてｙを、偶数奇数判定用にＰなる変数を一時的に使用する。まずｉ＝１とし（ステップ33）、ｉ＝＜Ｎである限り処理を繰り返すことにする（ステップ34）。次に第ｉの点から第ｉ＋１の点までを直線に結ぶ。これは第ｉの座標を（Xi、Yi）と書くことにしたとき、座標が（Xi、Yi）である点と（Xi＋１、Yi＋１）である点を、区間［Xi、Xi＋１］について線分の方程式

ｙ＝（Yi＋１−Yi）／（Xi＋１−Xi）＊ｘ＋Yi

で結ぶことに等しい（ステップ35）。但しｉ＝Ｎの時はｉ＋１＝１と見なし処理する。これをｉを１ずつ増加させ繰り返す（ステップ36）。ステップ35は実際には

ｘ＝（ｙ−Yi）＊（Xi＋１−Xi）／（Yi＋１−Yi）

という式を用いてｙ＝Yiからｙ＝Yi＋１までの区間で値を求め、水平方向の座標の位置が（ｘ÷８）の整数部分＋１で与えられる１バイト中の、（ｘ÷８）の余りの部分で示されるビットを１とする処理が行なわれる。垂直方向の変位はプリンタ構造体に示される解像度に従い決められた値を増分として与えられる。これを図14の線分49によって示す。但し図14、図15はいずれも閉じた図形の一部を示している。数学的な線分は太さを持たないが、53を０ビット目とするメモリ上の１バイト54を通過した部分にそれぞれ印刷すべきビット55を生じる。同様にして線分50、51、52が引かれた後ではメモリ上には、それら線分の通過した位置に印刷すべきビット55（図14の　）を生じる。

再び図13の流れ図に添って説明する。なおこれまでの

処理によってN個の座標の中から、水平垂直方向の最大値、最小値が求められておりそれらを、HMAX（水平方向の最大座標値）、HMIN（水平方向の最小座標値）、VMAX（垂直方向の最大座標値）、VMIN（垂直方向の最小座標値）と書き表す。すでに画像用のメモリの上には線分の位置に印刷すべき画素が示されているから、後はこれら画素により囲まれた部分を検出し、それらビットを１とすればよい。そのためにまず、一時的な変数Ｐを０にする（ステップ37）。次に垂直方向の変位を表す変数Ｖにその最大値VMAXを代入する（ステップ38）。それ以降の処理をＶ＞＝VMINであるかぎり繰り返す（ステップ39）。すなわち、水平方向の変位を示す変数HにHMINを代入し（ステップ40）、座標（Ｈ、Ｖ）で与えられる位置のビットが０であるか調べる（ステップ43）。ステップ43でビット＝１だったとき（Noの場合）、変数Ｐに１を加える（ステップ44）。変数Ｐの初期値が０であるから、図14の線分49で生じた印刷すべきビット（すなわち＝１であるビット）55にさしかかった時、変数Ｐはいつも１である。ステップ44を通った場合は、水平方向変位を１つ増し（ステップ48）、ステップ41で水平方向の最大値を超えていないか判断し、Ｈ＝＜HMAXであるかぎり、さらにステップ43以下の処理を繰り返す。もし水平方向の最大値を超えていたら、垂直方向の変位を１減じ（ステップ42）、前述の判断（ステップ39）に戻る。一方、再びステップ43の判断を通過するとき、例えば図13中の線分49と50の間の座標位置では（Ｈ、Ｖ）＝０であるからステップ45でＰが奇数かどうか判定され、Ｐが奇数であるために（Ｈ、Ｖ）＝１となるようにビットセットされる（ステップ47）。これを繰り返し、線分50の位置に達すると、ステップ43で（Ｈ、Ｖ）＝１であるからステップ44によってＰ＝Ｐ＋１という処理がなされる。このため線分50以降はＰが偶数であり、ステップ46を通過するため、ビット＝１とされない。この状態は線分51の位置で再度Ｐが奇数となるまで維持される。また線分52を通過した後はＰが偶数となる。

図14は線分49、50、51、52によって変数Ｐの偶数／奇数が繰り返され、塗りつぶしが行なわれた経過を図15のビット55（●）によって示す。従ってかかるアルゴリズムを実行した場合は、図11の多角形が塗りつぶされ、図12の図形を得ることになる。

以上の様な塗りつぶし手順を与えられた多角形に対して行なうのが「手続きFillPolygon」である。また「手続きInvertPolygon」は「塗りつぶしアルゴリズム」と同様な手順によって「線分に囲まれた領域」を偶数／奇数規則で、ビットの反転を行なう手続きである。この２つの手続きだけを用いて、次のようなプログラムを実行する場合を考える。

```
多角形Ａ
  ＝（6,
     (0,87)，  (22,93)，  (92,51)
     ，  (92,82)，  (20,27)，  (0,33)）
多角形Ｂ
  ＝（4,
     (34,58)，  (55,58)，  (55,34)
     ，  (34,34)）
多角形Ｃ
  ＝（4,
     (17,51)，  (78,51)，  (78,0)
     ，  (17,0)）
FillPolygon（↑多角形Ａ）
FillPolygon（↑多角形Ｂ）
InvertPolygon（↑多角形Ｃ）
```



上記のプログラム（説明のため実行可能な形式とは異なる）の行わんとする操作を図16を用いて説明する。上記プログラムはまず、６つの座標からなる多角形Ａ、４つの座標からなる多角形Ｂ、Ｃを定義する。これを図示すると多角形Ａは参照番号56、多角形Ｂは参照番号57、多角形Ｃは参照番号58の様に示される。但し多角形の座標を定義するために代入を行なった段階では、図16に示すごとに線分は現われない、図16は上記多角形Ａ、Ｂ、Ｃの形状を説明するための図である。続いて上記プログラムの手続きを次々に実行すると、図14の図形が得られる。すなわち、多角形56の領域を塗りつぶし、次に多角形57の領域を塗りつぶした後、多角形58の領域をビット反転する。この結果白ぬきされた領域が生じる。

ところで本発明を特徴付ける「分散処理」の考え方を誤って用いた場合、次のような事態が考えられる。図２の構成において、例えば、コンピュータ6aの分散処理インタープリタ２によって

```
多角形Ａ
  ＝（6,
     (0,87)，  (22,93)，  (92,51)
     ，  (92,82)，  (20,27)，  (0,33)）
多角形Ｂ
  ＝（4,
     (34,58)，  (55,58)，  (55,34)
     ，  (34,34)）
FillPolygon（↑多角形Ａ）
FillPolygon（↑多角形Ｂ）
```

までの処理を行ない、同時にコンピュータ6bの分散処理インタープリタ２を用いて

```
多角形Ｃ
  ＝（4,
     (17,51)，  (78,51)，  (78,0)
     ，  (17,0)）
InvertPolygon（↑多角形Ｃ）
```

という処理を行ったものとし、それら処理の結果をコンピュータ6aの分散処理インタープリタ２で単純に重ね合わせたものとすれば、得られる結果は図18のようなものにしかならない。上記のプログラムは後に続く手続き

が、先に登場した手続きの結果を受けて処理するものであるから当然である。

これはインタープリタを分散処理しようとした時、常に発生する問題である。一つの解決方法は、PDLの文法に強い制約を受け、先の手続きの結果を参照しない言語仕様とすることであるが、この方法はアプリケーションソフトのコーディングにおいて負担を増すため、本実施例では採用しない。本実施例は別の手段によってこの点に解決を与えた。すなわち、プログラム言語BASICなどに用いられる、計算／汎用プログラムを対象としたインタープリタと異なり、PDLのインタープリタは文字、図形を画素に変換する操作が対象となる、このことは複数個の文字／図形が重なり合って形成する１つの図形群（グループ）が、１ページの印刷用紙に相当する空間に複数個ある場合、個々の「図形グループ」が空間上で離れて存在すれば、それらを「切り離されて存在する一つ一つのグループ」ごとに画素に置き換えることは可能であることを示している。そこで本発明は「切り離しても他の図形に影響を及ぼさない図形グループ」を検出する手段を考案し、PDL並列処理インタープリタに用いることにした。以下にその例を示し、説明する。

以下に説明する幾つかの例に共通して言えることは、図形どうしの重なり合いを、実際に画素に変換する以前の状態でいかに早く検出するかという点である。画素への置き換えを分散処理することが目的であるから、実際に画素が生成された後に、重なり合いが検出されるのでは意味がないためである。以下の例では多角形、楕円などを用いて説明を行なうが、先に示したPDLの仕様あるいは、類似の仕様であらかじめ記述されたコードが存在し、従って多角形の座標についてはその値が本実施例の処理以前に既知となっている、（すなわち、その値を取り出して参照することが可能である。）ものとする。

（１）　グループ判別の例１

図19、20及び21を参照して説明する。ここで、PDLによって書かれたソースコードの意図するところが、例えば

手順1:長方形の領域に操作を加える。

手順2:三角形の領域に操作を加える。

手順3:楕円の領域に操作を加える。

というものであると仮定する。ここで言うところの「操作」とは「塗りつぶし、ビット反転、ビット和」などの何らかの操作であるとする。

まず長方形61の領域に操作を加え、次に三角形62の領域に操作を加え、最後に楕円65の領域に操作を行ないたいものとして、それらの位置関係は図20の通りである。人間の視覚では、明らかに長方形61と三角形62が重なっている。それを検出する方法が本例である。

すでに説明したように、本発明の分散処理インタープリタはPDLによって記述された、文書／図形記述のためのソースコードを翻訳する過程で、一度中間コード形式のファイル10を作り、主記憶、あるいは補助記憶に記録し、実際の分散処理、画素への変換はこの中間コード形式のファイルを参照しつつ行なう。かかる中間コードの解析によって、長方形61の座標が明かとなると、その「左上端の点」の座標の水平、垂直位置の値をそれぞれ、変数のペア（GLTH［１］、GLTV［１］）に代入する。また「右下端の点」の座標の水平、垂直位置の値をそれぞれ、変数のペア（GRBH［１］、GRBV［１］）に代入する。但し、ここで水平、垂直とは図19を正面視した状態での方向を差すものとし、その正負は、図19の矢印によって示すものとする。また１ページの文書中には、複数の図形グループが存在するため、変数、GLTH、GLTV、GRBH、GRBVはいずれも配列変数であり、例えば、GLTH［１］は変数GLTHの配列の第１要素であることを示す。次に三角形62の３つの座標が明かとなるため、それら座標の値より、水平方向の最大値Hmax、最小値Hmin及び、垂直方向の最大値Vmax、最小値Vminを検出する。これは、左上端の座標が（Hmin、Vmax）であり、右下端の座標が（Hmax、Vmin）であるような長方形領域63を見つけることに等しい。これを図21にステップ67として示す。

次に長方形61と63の重なりを検出する。まずVmax≧GRBVであるかどうか調べる（ステップ68）。もし長方形63の上端の垂直位置（Vmax）が長方形61の下端の垂直位置（GRBV）未満であれば、長方形63は長方形61の下方に位置し、重なりは生じない（ステップ75）。そうでない場合、Vmax≦GLTVであるかどうか調べ（ステップ69）、これが不成立かつ、Vmin≦GLTVの条件（ステップ70）も不成立であれば、長方形63は長方形61の上方に位置し、重なりは生じない（ステップ75）。それ以外の場合は、水平方向の位置関係を調査する。すなわち、Hmin≦GRBHであるか調べ（ステップ71）、不成立であれば、長方形63は長方形61の右に位置し、重なりを生じない（ステップ75）。ステップ71の条件が成立しても、Hmin≧GLTHの条件（ステップ72）が不成立で、Hmax≧GLTHの条件（ステップ73）も不成立であれば、長方形63は長方形61の左に位置して、重なりは生じない（ステップ75）。

以上の場合以外の場合、重なりが検出される（ステップ74）。図19の例では、長方形61と63の重なり合いが検出される。この結果長方形61と三角形62は「１つの図形グループ」に属していると判断する。

また、これにより従来長方形61が属していた「図形グループ１」に三角形62も加わったわけであるから、「図形グループ１」の領域も更新しなければならない。すなわち

手順1:変数GLTH［１］にGLTH［１］またはHminの内、どちらかより小さい方を代入する。

手順2:変数GLTV［１］にGLTV［１］またはVmaxの内、どちらかより大さい方を代入する。

手順3:変数GRBH［１］にGRBH［１］またはHmaxの内、ど

ちらかより大さい方を代入する。
手順4:変数GRBH［1］にGRBH［1］またはVminの内、どちらかより小さい方を代入する。
という手順を行なう。これにより、「図形グループ1」の新しい領域は、図20の長方形64のようになる。ここにさらに楕円65に対しての操作が加わるが、これは楕円の方程式から楕円65が内接する長方形66の各座標が求められるため、新たな（Hmin、Vmax）、（Hmax、Vmin）の値を代入し、再び図21の流れ図にしたがって、判断を行なう。その結果、領域66は領域64と重なり合いを持たないことが知れるため、領域66は楕円65が含まれる「新たな図形グループ」の境界を示す領域であることがわかる。これを「図形グループ2」として
GLTH［2］=Hmin
GLTV［2］=Vmax
GRBH［2］=Hmax
GRBV［2］=Vmin
という4つの代入が行なわれる。

一般の文書はより複雑であるが、手順は全く等しい。

以上の様な処理を開始する以前に、文書を記述したソースファイルは閉じており、（すなわちページ開始からページ終了までの記述が完結していて、これ以上追加の書き加えが無い状態になっている。）さらに中間コードへの変換も終了しているため、図形要素／文字は有限個であると言う保証がある。いますでにN個まで図形グループが検出されたものとして、その次に現われた図形要素の扱いを決める手順の流れ図を図22に示す。図22中で変数iは図形グループを数えるための整数、変数jは注目している図形要素が含まれる図形グループの数を示す整数、配列変数Groups［j］は注目している図形要素が含まれる図形グループ（複数であることが考えられる。）が第何グループであるか記録するための整数配列である。

まずi、jを初期化し（ステップ76）、図21の流れ図に示した処理をサブルーチンとして呼び出して重なりを検出する（ステップ77）。ステップ78で重なりが無しである場合は、iを1だけ増し（ステップ86）、全ての図形グループについて終了していない場合は（ステップ81）、次のグループについて重なり検出（ステップ77）を繰り返す。一方、ステップ78で重なり有りの場合は、以下の手順
手順1:変数GLTH［i］にGLTH［i］またはHminの内、どちらかより小さい方を代入する。
手順2:変数GLTV［i］にGLTV［i］またはVmaxの内、どちらかより大さい方を代入する。
手順3:変数GRBH［i］にGRBH［i］またはHmaxの内、どちらかより大さい方を代入する。
手順4:変数GRBH［i］にGRBH［i］またはVminの内、どちらかより小さい方を代入する。
で図形グループiの領域を更新し（ステップ79）、重なりが検出された回数を示す変数jに1を加え、重なりが検出された図形グループの番号（ここではi）を図形グループの配列Groups［j］に代入し、iについても1を加える（ステップ80）。

従って、例えば、注目の図形が3つの図形グループと重なり合いを持ち、それら図形グループが、14番目、17番目、22番目であるとすれば、
Groups［1］=14
Groups［2］=17
Groups［3］=22
という配列が得られ、この時j=3となっている。

ステップ81でN個の図形グループ全てについて検査が終わったと判断されると、j=0であるか検査される（ステップ82）。j=0であれば重なりは存在しなかったことになるから、注目していた図形要素は、新しい図形グループを作ることになる。従って新しい図形グループの生成（ステップ83）が行なわれる。すなわち、
N=N+1
GLTH［N］=Hmin
GLTV［N］=Vmax
GRBH［N］=Hmax
GRBV［N］=Vmin
という5つの代入がなされる。j=0でない場合は、j=1であるか判断し（ステップ84）、これが真であれば、グループの更新は既にステップ79で終了しているので、何も行なわない。

それ以外の場合は、jは2以上の整数である。つまり2つ以上の図形グループと重なり合いを持っている。この場合は、図形グループ間の結合と新しい領域の設定（ステップ85）が必要である。これを図23および24を参照して説明する。図23は図形61、62からなる図形グループと、図形65からなる図形グループが既に検出されているものとして、現在注目している多角形87が前記2グループのどちらにも重なり合いをもっている場合を示す図である。従って図22の流れ図で言い表せば、ステップ85の前段階で、j=2、Groups［1］=1、Groups［2］=2である。ステップ85は新たに、図形61、62、65、87からなる図形グループを作り、その領域を破線で囲んだ長方形領域88のように設定する。

より一般的には、「図形グループp」と「図形グループq」に現在処理中の図形要素が重なりを持っている場合、ステップ85では以下の手順を行なう。但しp<qとする。
手順1:変数GLTH［p］にGLTH［p］、GLTH［q］またはHminの内、どちらかより小さい方を代入する。
手順2:変数GLTV［p］にGLTV［p］、GLTV［q］またはVmaxの内、どちらかより大さい方を代入する。
手順3:変数GRBH［p］にGRBH［p］、GRBH［q］またはHmaxの内、どちらかより大さい方を代入する。
手順4:変数GRBV［p］にGRBV［p］、GRBV［q］または

31

Vminの内、どちらかより小さい方を代入する。

手順5:GLTH［q］＝－32768

GLTV［q］＝－32768

GRBH［q］＝－32768

GRBV［q］＝－32768

という代入を行なう。

当然、３つ以上の図形グループを結合する場合も手順は同様である。また従来図形グループqが使用していた配列要素に－32768を代入するのは、次回以降の処理において、いかなる図形とも重なり合いを生じさせないためであり、また新たな図形要素が発生した場合は、配列要素の値が－32768であるような配列は未使用であるとものと検出し該配列への代入を認めるためである。なおこの場合図形グループの総数を示す整数Ｎは更新を必要としないのは言うまでもない。

以上のようにして全ての図形要素についてグループ分けが完了する。文字についても扱いは全く同じであり、その形状、ポイント数によって図形として占める大きさも事前に知れるため、それらを取り囲む長方形領域を設定し、他の図形要素と同じく取り扱う。

本例のグループ判別では、図24の図形グループ領域88のごとく、これに含まれる図形要素に比較し、図形グループ領域88が大きすぎる様に見える。しかし実際には検出された個々の図形グループは、それぞれ別個のプロセスで動作中の分散処理インタープリタの塗りつぶしアルゴリズムによって処理され、それら塗りつぶし手段は前述の様に水平方向を走査方向としたものであるから、処理が特に遅延することはない。もちろん行なわれる操作が塗りつぶしではなく、ビット反転などのビット操作であっても同様である。

（２）　グループ判別の例２

次にグループ判別の別の例について説明する。図25、図26に本例の説明図を示す。図中、参照番号89は印刷用紙を想定した領域を示し、破線90は印刷可能領域を縦11分割、横８分割した小領域を示している。これら個々の分割された小領域は次のように定義される値ｒにより一意に指定される。すなわちｉ列ｊ行目の領域であればこれをＲ（ｉ、ｊ）と書き表し、連続した値

ｒ＝（ｊ－１）×８＋ｉ

は領域Ｒ（ｉ、ｊ）を指し示すために用いる。

図25では、図形要素91、92、93、94が印字領域上に存在し、この内、長方形91と三角形92が重なりを持っている。従って図形要素91、92は同一の図形グループに属しているものと判断しなければならない。本例では、注目している図形要素についてまずその図形要素の境界を指定する座標が、破線90によって分割された領域群の内どこに含まれるかを検出する。印刷可能領域の左下の角の点を原点とし、水平方向については右を＋方向、垂直方向については上を＋方向とすれば、領域Ｒ（ｉ、ｊ）の左上端の座標はGLTH［ｒ］＝（ｉ－１）・hGLTV［ｒ］

32

＝ｊ・ｖと書き表され、右下端の座標はGRBH［ｒ］＝ｉ・hGRBV［ｒ］＝（ｊ－１）・ｖとなる。ここでｈ、ｖはそれぞれ水平、垂直方向の解像度から決まる定数であり、例えば、横方向の印刷可能領域を８インチ、縦方向の印刷可能領域を11インチとしてどちらも印刷画素の解像度を300画素／インチとすれば、ｈ＝300×８インチ÷８区分＝300v＝300×11インチ÷11区分＝300である。またｒは前述の指標である。これを用いて現在注目している図形要素の境界のうち一つの角の点の座標を（ｘ、ｙ）とすれば、GLTH［ｒ］≦ｘ≦GRBH［ｒ］かつGRBV［ｒ］≦ｙ≦GLTV［ｒ］であるかどうかをｒ＝１からｒ＝88まで順次調べるだけで良い。

このような検出を行えば、領域91の４つの座標は斜線で示される領域95にそれぞれ含まれ、また領域92を定める３つの座標はそれぞれ横線で示される領域96に含まれることが検出される。例えばここでは領域96はＲ（７、６）、Ｒ（４、７）、Ｒ（６、９）の３つである。

次にこれら領域の指標のうち水平方向での最小値、最大値、垂直方向での最小値、最大値を求める。領域96の場合これらはそれぞれ、４、７、６、９となる。そしてこの範囲に指標が含まれる様な領域を全て取り出す。それらはＲ（４、６）、Ｒ（５、６）、Ｒ（６、６）、Ｒ（７、６）、Ｒ（４、７）、Ｒ（５、７）、Ｒ（６、７）、Ｒ（７、７）、Ｒ（４、８）、Ｒ（５、８）、Ｒ（６、８）、Ｒ（７、８）、Ｒ（４、９）、Ｒ（５、９）、Ｒ（６、９）、Ｒ（７、９）である。指標ｒを使って言えば、集合｛r|44、45、46、47、52、53、54、55、60、61、62、63、68、69、70、71｝であるような領域である。これらを全ての図形要素について求め、図形要素ごとに配列として記録しておく。但し文字については１文字を１図形要素とするのではなく、複数の文字列からなる区分を作り、それを１図形要素と考える。

この結果、図形要素91と92についてみると指標ｒの値が、ｒ＝60、ｒ＝61、ｒ＝68、ｒ＝69の部分で共通であることが判明する。その結果図形要素91、92には重なり合った部分があるものと判断し、先に記録した配列を参照して全領域の和集合を作ると、図26の斜線部97の領域群を取り出すことが出来る。すなわち斜線部97が、91、92の両図形要素を含む図形グループである。同様にして領域群98、99などの図形グループが検出される。

整理してより一般的に手順を書くと、

手順1:図形要素ｉについて、その境界を指定する多角形の１つ１つの座標につき、その座標が含まれる領域を求める。これをＲ（H1、V1）、Ｒ（H2、V2）、…Ｒ（Hn、Vn）とする。

手順2:上記H1からHnまでの内から最小値Hmin、最大値Hmaxを求める。同じくV1からVnまでの内の最小値Vmin、最大値Vmaxを求める。

手順3:領域Ｒ（Hmin、Vmin）を左下角、領域Ｒ（Hmax、Vmax）を右上角とする矩形領域に含まれる全ての領域Ｒ

（ｉ、ｊ）の指標ｒを求め図形要素ｉについての配列Unit［ｉ、ｒ］に書き込む。（配列の要素は1/0とすれば、Unit［ｉ、61］＝１、Unit［ｉ、62］＝０などの代入を行なう。）

手順4:全ての図形要素について手順１から３を繰り返す。

手順5:手順３で作られた全ての配列の重なりを検出し、重なりが１つ以上検出された配列は論理和をとる。

手順6:手順５によって重複した配列は削除し、残った配列を図形要素からなる図形グループの領域を与える領域Ｒ（ｉ、ｊ）の集合として、採用する。

以上の様な手順でグループ判別が行なわれる。この方法では図26の領域群97、98のように前述の第１の実施例では同一図形グループとして判断されてしまう様な図形も別グループとして判別出来るため、より分散処理に適している。

（３）　グループ判別の例３

図形グループ判別の第３の例を図27を用いて説明する。一般に多くの文書は、横方向に一連の文字／図形などのつながりを持つ場合が多い。従って、前例のような縦横両方向の領域分割ではなく、より単純な横長の矩形のみへの分割を採用しても充分実用的なグループ分割が可能であり、実行速度の点ではむしろ有利であるとも言える。本例はこの点に注目したものである。

図27において、直線89で囲まれた領域は印刷用紙の範囲を示し、破線100は印刷可能領域を縦方向に11分割した場合の境界を示し、参照番号106は分割された個々の領域を示す。印刷可能領域の左下の角の点を座標原点として、水平には右方向を＋、垂直には上方を＋とする様な、座標系を設定する。１つの領域をＲ（ｉ）と書き表せば、領域Ｒ（ｉ）の左上端の座標は（０、ｉ・ｖ）であり、右下端の座標は（Ｗ・ｈ、（ｉ－１）・ｖ）である。但しＷは印刷領域の幅であり、ｈ、ｖはそれぞれ、

ｈ＝印刷密度（単位　画素／インチ）

ｖ＝印刷密度×印刷領域の高さ÷分割数

である。

グループ判別の手順は

手順1:注目している図形要素について、その図形要素の境界を与える座標が含まれる領域Ｒ（ｉ）を見つける。その指標ｉを図形要素ごとに与えられた配列Unit［ｊ、ｉ］に書き込む。１≦ｉ≦11であるから、図形要素の総数を32767個以下などとすれば２次元配列Unitは特に大きな配列とはならない。

手順2:全ての図形要素についてその領域を定める全ての座標を対象として、手順１を繰り返す。

手順3:手順２までででで作られた全ての配列の重なりを検出し、重なりが１つ以上検出された配列は論理和をとる。

手順4:手順３によって重複した配列は削除し、残った配列を図形要素からなる図形グループの領域を与える領域Ｒ（ｉ）の集合として、採用する。

これを図27の図形要素101、102、103、104、105を例にとり説明する。前記101から105までの図形要素をそれぞれ図形要素指標で１から５とすれば、

Unit［1,9］＝1,Unit［1,10］＝1,Unit［1,11］＝1,Unit［1,1］＝0,Unit［1,2］＝0,,,Unit［1,8］＝０

Unit［2,8］＝1,Unit［2,9］＝1,Unit［2,1］＝0,,Unit［2,10］＝0,Unit［2,11］＝０

などの結果が得られる。手順３によって図形要素101と102には重なりが検出されるため、それらは同一の図形グループと判断され、従って領域Ｒ（11）、Ｒ（10）、Ｒ（９）、Ｒ（８）は１つの図形グループの領域と判断される。同様にして図形要素103、104、105も同一の図形グループを構成するものと検出される。

このグループ検出では、図形要素101、102の様に実際には重なりを生じていない図形要素（従って分散処理可能である）であっても、同一の図形グループであるかの様に認識するため、若干冗長ではあるが、簡単に実行できる方法である。

以上の３つの例の示したごとく相互に重なり合いを生じていない図形群を捜しだし、図形グループとして区別することが可能である。実際に様々なアプリケーションソフトウェアが作りだすドキュメント（文字／図形を含む文書）について図形グループを検出してみると数個から数100グループが検出され、1000を超えることは稀である（但し文字については、１行の中で同一字体であるかぎり、できるだけ文字列としてグループ化するものとして扱った）。実際には分散処理に使用できるコンピュータが３台、図形グループの数が400個、などという場合が多いものと考えられる。また１台のコンピュータに関しても特に処理能力に余裕があれば、本発明の分散処理のために生成するプロセスを１つに制限する必要も特にない。いずれにしても検出した図形グループの数と生成される本実施例の用いるプロセス数は一致しない。また分散処理の一部を分担するコンピュータにとっては、大きな負荷のかかる処理に長時間参照するのは好ましくないのは明らかである。なぜなら要求の発生した時点で偶然プロセッサの負荷が小さかったとしても、次の瞬間には大きな負荷を伴う処理要求が発生するのは充分考えられることだからである。これを回避する良い方法の１つは出来るだけ小さい部分を処理させ、処理の終了ごとにプロセッサの負荷を調べ直すことである。本発明のPDL並列処理インタープリタではこれを実施するために以下に示す表管理を採用する。

本発明のPDL並列処理インタープリタではクライアントプロセスの管理する記憶領域上に表３に示すような、「図形グループ管理表」を作り１ページの全ての印刷が終了するまで管理する。図４を参照して既に説明した様に、グループ管理表217は、クライアントプロセス210の実行時に、グループ検出処理216及び画像合成処理220に

より参照される。

表３

| 図形グループ | 図形グループの領域 | 処理状況 | 送り先 |
|---|---|---|---|
| 1 | GLTH[1], GLTV[1], GRBH[1], GRBV[1] | -1, -1 | 6a |
| 2 | GLTH[2], GLTV[2], GRBH[2], GRBV[2] | 100, 140 | 6b |
| 3 | | 0, 0 | 6b |
| N | GLTH[N], GLTV[N], GRBH[N], GRBV[N] | -1, -1 | 6a |

この図形グループ管理表は

（１）図形グループへの指標：整数１から512まで

（２）図形グループの印刷用紙上に占める領域:4つの整数の組　領域は全て矩形に取り、該図形グループの左上端の水平座標（GLTH［ｉ］）、左上端の垂直座標（GLTV［ｉ］）、右下端の水平座標（GRBH［ｉ］）、右下端の垂直座標（GRBV［ｉ］）の順に16ビット長で指定する。

（３）処理状況:2つの整数の組－１、－１の場合未処理、０、０の場合処理済みかつクライアントプロセスへ転送済み、正の整数の組の場合、必ず“第１の数＜第２の数”であり、垂直座標が「第１の数の位置」から「第２の数の位置」まで画素に変換済みを意味する。

（４）転送先のコンピュータのネットワークでの名称：オペレーティングシステムの認める長さの文字列。

という４つのメンバーからなる表である。前述のグループ判別の第２例によると図形グループの領域は矩形とはならないが、表管理の上では最大最小となる水平／垂直位置を用いて矩形領域として管理する。

以下図28を参照して、図形グループ管理表を用いた管理を具体的に説明する。

印刷用紙の領域を138に示し、その中に斜線部141、142、144などで示される図形グループの領域があるものとする。印刷装置は紙の排出方向に対し直交する方向の帯状の矩形をとり印刷するのが好都合である場合が多い。図28では、例えば直線139、140に挟まれた領域の様な部分ごとに印刷する好都合である。これは印刷装置の方式に依存するがいずれにしても、印刷用紙全体に必要な全ての画素を１度に展開して保持するだけの記憶領域を印刷装置が持てば、操作は最も単純である。しかし印刷装置のコスト等の制約から必要なメモリの一部しか実装できない場合は、区分区分ごとに印刷することになる。このためクライアントプロセスには、印刷装置が現在印刷しようとしている部分についての画素を帯状に並べて印刷装置に転送することが要求される。いま直線139から140までの領域を印刷しようとする場合、クライアントプロセスは表３の仕様に従いメモリ上に作った表から図形グループ141、142、143のそれぞれの画素への変換状況を調べる。直線139の垂直座標を1200、直線1400の垂直座標を1500とすれば、図形グループ143の画素への変換が少なくとも垂直座標で1500まで終了していなくてはならない。これは表中の第３の要素から知ることが出来る。クライアントプロセスは印刷の実行にあたり「印刷装置への画素の転送」と「表の更新」という少なくとも２つのプロセスを同時に実行（並列処理）しなければならない。前者の図29の流れ図に、後者を図30の流れ図に示す。

以下この２つのプロセスを説明する。まず印刷すべき帯状の領域が設定される（ステップ145）。これは図28では例えば垂直139、140に挟まれた領域である。次に上記印刷領域に含まれる図形グループの検出がグループ管理表の第２要素に基づき検出される（ステップ146）。印刷すべき領域について全ての図形グループが展開済みであるか調べる。グループ管理表の第３要素が負数または０でなければ展開が進行していることを意味し、この第３要素により印刷すべき領域より先まで展開が進んでいるかを調べる。図28の例で示せば、図形グループ141、142の展開が完全に終了していることと、図形グループ143の処理が垂直方向に1500まで進んでいることが必

要である。印刷すべき領域について全図形グループが展開済みであれば（ステップ142）、印刷領域の転送要求をサーバプロセスに対し行なう（ステップ148）。この場合、図形グループの全領域ではなく、印刷に必要な領域のみについて画素を転送するようにサーバプロセスに要求し、おして転送された画素を受信する（ステップ149）。受信した画素は印刷装置に転送し、図形グループの全ての領域について印刷が終了したものに関してはグループ管理表の第３要素を０とする（ステップ150）。以降はページ終了まで印刷領域を更新しつつ繰り返し処理する（ステップ151）。

これと並行して、クライアントプロセスの管理下で、グループ管理表の更新が行なわれている（図30）。これは次のような処理である。まず初期化のため全図形グループの処理状況を－１とし（ステップ152）、図形グループを示す指標ｉも初期化する（ステップ153）。以下図形グループｉの展開を担当するコンピュータのネットワーク上での位置をグループ管理表の第４要素から求め、状態要求のメッセージを送り、図形グループｉ画素への変換の状況を知る（ステップ156）。これに基づき表を更新し（ステップ157）、次の図形グループへ移る（ステップ158）。ステップ154で一連の全図形グループについて同様の調査を行ない、全グループ終了したら、処理を一旦終了する。単位時間経過後には再びステップ153から開始する。この間の時間経過で既に印刷された図形グループに関しては、第３要素＝０、０であることから検出できるため（ステック155）、メッセージの授受はせず、その他の図形グループについて上記手順と同様に印刷画素への展開状況を調べ（ステップ156）、表を更新する（ステップ157）。

以上の様なグループ管理表の操作により本発明では分散処理の進行状況を管理する。

次に本発明のPDL並列処理インタープリタの第２の実施例について述べる。

既に第１の実施例において示したように、本発明の分散処理インタープリタの実行時における役割は、大きく分けて２つである。それは

（Ａ）PDLソースファイルを分割し分散処理を行なっても、これを単一の翻訳系で処理した結果と異なること無きように、分割可能な図形グループを検出し、他の情報機器に分割したコードを送り付ける、

（Ｂ）実際にPDLソースファイルを翻訳し、印刷可能な画素に置き換えること、

である。

前記第１の実施例は、１つの処理系が上記（Ａ）、（Ｂ）いずれの手段も実行可能であり、その時にサーバとなるかクライアントとなるかだけが役割を区別するものであった。こちろん役割を常に固定した複数個のプログラムからなる分散処理系を構成することも可能であり、本実施例はそれを行なったものである。以下、図31を参照して説明する。

本実施例では、印刷装置108に上記（Ａ）の処理を実行するプログラム107が常駐しており、ネットワーク７によって接続されたコンピュータ6a、6b、6cには上記処理（Ｂ）を実行するプログラム106が存在する。コンピュータ6a、6b、6cのいずれかで実行中のアプリケーションソフトウェアにおいて、印刷要求が発生した場合を考える。ここでは説明のためコンピュータ6aで実行されていたアプリケーションソフトウェア（例えばワードプロセッサ等）を使用していたユーザが、印刷処理を要求したとする。

以下、図32の流れ図に従って説明する。印刷処理を行なおうとするアプリケーションソフトウェアはまず、印刷処理の対象となる複数ページ（もちろん１ページでも良い。）の中から、現在処理の対象としているページを印刷するために、そのページを開く（ステップ109）。言い換えると印刷処理に必要なだけの記憶領域を主記憶装置又は補助記憶装置のいずれかに確保し、フォント情報など必要なデータを準備する。そして実際に文字や図形等の図形要素をPDLの仕様に従い記述する（ステップ110）。処理するページに対する全ての書き込みが終了したら、ページを閉じる（ステップ111）。これはページ終了記号を書き込んだ後、ステップ109が確保したメモリ領域を書き込み禁止とする処理である。ここでステップ110の書き込み処理が主記憶装置で行なわれた場合は、それら書き込み内容をファイルに書き直し、また補助記憶装置で前記書き込み処理が行なわれた場合はファイル先頭に適当な識別子を付加することで印刷ファイルを生成する（ステップ112）。そして該ファイルを印刷装置108に転送（ステップ113）すれば、１ページあたりの印刷処理は終了する。この後アプリケーションソフトウェアは次ページの印刷処理あるいは、その他の処理に移る。

以下の処理は印刷装置108上のプログラム107が行なう。印刷装置108がPDLのソースファイルを受信する場合と印刷すべき画素のデータを受信する信号とがあるが、ここで述べるのは前者の場合である。まずステップ114で識別コードを受信し、それを判別する。次にPDLで記述されたソースファイルを受信し（ステップ115）、処理に適した中間コード形式に翻訳する（ステップ116）。具体的には手続きFillPolygon（^PolygonA）等のように文字コードの列で記述されたソースコードから、16ビット長の作用素（オペレータ）と32ビット長の作用子（オペランド）の並びに置き換える。例えば、16進数でFillPolygon＝＄C044, InvertPolygon＝＄C084などと置き換える。もちろんこの様な中間コードにどの様なビット長のどの様なコードを使用するかは処理系の任意である。

次に既に述べたいくつかの方法、または類似の方法によって分散処理を行なうことの出来ない（分割不可であ

る）図形グループを検出し（ステップ117）、中間コードファイルを各図形グループに分割する（ステップ118）。この段階で分割された各図形グループは、それぞれ完全に独立した別個の処理として行ない、後に任意の順番で重ね合わせを行なっても、正しい印刷結果が得られるという保証が得られている。次いで、ネットワーク上に存在する他の情報処理機器（主としてコンピュータ）に対しプロセスの生成を要求し（ステップ119）、応答の有った機器に対しては、中間コード化された前記図形グループを転送する（ステップ120）。この転送は前述のスケジューリングの手順に従って行う。この後、印刷装置108上のプログラム107は印刷データ受信待ちに入る。

一方、ステップ119のプロセス生成要求を受けて動作するのはコンピュータ6a, 6b, 6c内の各プログラム106である。プログラム106には常駐部分と非常駐部分が存在し、常駐部分は常にネットワーク上で発生する要求を検出しており、非常駐部分は必要な場合のみメモリが確保され呼び出されて実行されるプログラム単位（プロセス）として管理される。

プログラム106、107の流れ図を図33に示す。プロセス生成要求を受信したプログラム106の常駐部分はまずメモリを確保する（ステップ121）。これが可能であれば非常駐部分を該メモリ領域の一部に読み込み、プロセス生成を行なう（ステップ122）。次に印刷装置108のプログラム106から送信された中間コードを受信し（ステップ123）、これを翻訳し実際の印刷画素に置き換える（ステップ124）。この印刷画素生成の手順は第１の実施例で説明したとおりであり、翻訳は１ページ中の記述の内の何分の一の分量に過ぎないため、展開に使用するメモリ、実行時間共に軽減される。作りだされた画素は再び印刷装置108のプログラム107へ転送される（ステップ125）。

プログラム107はこの画素データを受信すると識別コードによって、画素データ列であることを認識し（ステップ126）、画素データを受信し印刷用のバッファー領域にて整列させる（ステップ127）。該バッファーが充足した場合、印刷（ステップ129）を行なう。もし１回のバッファーの充填で１ページ全部のデータに満たない場合は、１ページのデータ全ての印刷が終了するまで処理を繰り返す（ステップ130）。１ページの処理が終了した場合は紙を排出し（ステップ131）、１ページあたりの印刷処理を終了する。

印刷装置の印刷方式によっては、上記手順は異なる場合がある。周知のごとく電子写真方式の印刷装置は一連の印刷プロセスが開始されてから終了するまで、プロセスの中断が出来ない場合が多い。このプロセスの進行に合わせてネットワークからデータを転送し続けるのは、現在の技術では困難である。この場合はステップ127、128で１ページのデータに必要なだけのメモリを確保する必要がある。しかしいわゆるシリアルプリンタを上記手順に用いた場合には、印刷途中で中断することはいかなる時点でも可能であり、バッファー領域は１ページ分必要ではない。こうした制約は主に印刷装置の物理的な制限によるもので、バッファー領域としてどの程度の大きさのメモリを実装できるかというハードウェアの制約は、分散処理を行なうことには制限を加えるものではない。本発明は上記のどちらの場合にも対応できる。

次に本発明の第３の実施例を説明する。

印刷時のバッファーとして用いるメモリ領域、文字印刷に使用する書体のデータ等は、ネットワーク上に分散して置かれても良いが、１つの情報機器に集約することも当然可能である。言い換えると、

（１）　PDLのソースコードを翻訳する処理は分散する、

（２）　印刷用バッファー領域、フォントデータ、カラー印刷時の色変換テーブル等、ソフトウェアに必要な資源はネットワーク上の１箇所に保持する、という形態の分散処理系も可能である。図34にこの様な系を実現する１つの実施例の構成図を示す。ネットワーク７によってコンピュータ6a、6b、6cが接続されているほか、装置133も接続されている。装置133は表示用のCRTなどディスプレイに類する装置を付属していない点を除いて、内部のハードウェアの構成はコンピュータ6a、6b、6cと全く等しい。装置133はインターフェース137により補助記憶装置（ハードディスク等）134に接続されているほか、別のインターフェース136によって、印刷装置135に接続されている。ここでは印刷装置135で使用する印刷方式として電子写真方式を用いるものとし、装置133には１ページの印刷に必要な印刷画素を全て記録するに充分な容量のメモリが実装されており、インターフェース136は一連の電子写真印刷プロセスの１ページの印刷時間内に必要なデータを全て転送できる構成である（現状の技術でSCSIインターフェース等の高速パラレルインターフェースが４メガバイト／秒程度の転送速度を有し、この構成で使用可能であり、高速ビデオインターフェースも同様に使用可能である）。また、補助記憶装置134には各種字体の外形形状を記述したフォントデータが記録されており、装置133を介してネットワーク上の他の情報機器から該データを読みだし可能である。

以上の構成においてコンピュータ6a、6b、6cにはPDLソースファイルを翻訳して印刷可能な画素に置き換える処理を実行するソフトウェア106が組み込まれており、ソフトウェア106はその常駐部が常に動作をしており、ネットワークからのメッセージに応じて非常駐部を呼び出し実行する構成である。一方装置133にはソフトウェア132が組み込まれており、該ソフトウェアはネットワーク上の他の情報機器から見た時、仮想された印刷装置のごとく振る舞うものである。今コンピュータ6aの上で動作している何らかのアプリケーションソフトウェアか

ら印刷要求が発生したものとして、本実施例の動作を説明する。

手順1:発生した印刷要求はネットワーク７を介してソフトウェア132により受信される。

手順2:ソフトウェア132はページ開始からページ終了までの全てのページ記述のコードを受信する。従って、他の情報機器は装置133が印刷装置であるかのように錯覚するわけである。

手順3:ソフトウェア132は手順２で受け取ったソースコードから中間コードを生成する。該中間コードから分割可能な図形グループを検出し、グループごと識別子を付加し表に記録する。

手順4:132はネットワーク７を介し分散処理に応じられる情報機器を捜すためのメッセージを送信する。すなわち、プロセス生成要求を行なう。

手順5:ソフトウェア106の常駐部のうちこれに応じられるものは、メモリを確保し、非常駐部を呼び出しプロセスを生成し、応答をソフトウェア132に返す。

手順6:ソフトウェア132は応答の有った機器に対し手順３で作った表に従い、あらかじめ分割しておいた図形グループの記述を行なった中間コードを、手順５によって作りだされたタスクに送信する。

手順7:前記プロセスによって翻訳された印刷画素は、ソフトウェア132の要求に応じて装置133に転送され、装置133の印刷用バッファー領域に展開される。

手順8:上記バッファーに１ページに必要な全ての印刷画素が整列したら、これをインターフェース136によって印刷装置135に転送し、実際の印刷を行なう。以上の手順によって本発明のPDLの分散処理による翻訳が実施される。

次に、図35を参照して第４の実施例を説明する。

本実施例においてはネットワーク７には複数台のコンピュータ6a、6b、6cの他、複数台の印刷装置159a、159b、159cが接続されている。この環境において本発明のPDL並列処理インタープリタは印刷装置159a、159b、159c上にソフトウェア160としてのみ組み込まれている。ソフトウェア160はこれまでの実施例と同様に、「常駐部分」、「非常駐部分」から構成されている。常駐部分の行なう処理は、ネットワーク上に発生する様々な要求のメッセージを受信し、自己に対する印刷要求であれば、非常駐部分の内の「クラインアントプロセス」となる部分を呼び出し実行し、他の分散処理インタープリタからのプロセス生成要求であれば、非常駐部分の内の「サーバプロセス」となる部分を呼び出す、という２項目だけである。ここで言うクライアントプロセス、サーバプロセスの役割も第１の実施例の述べたところに等しい。すなわちクライアントプロセスは、

（１）　PDLソースコードによって書かれた文書の記述を中間コードの段階まで翻訳し、

（２）　分割可能な図形グループを検出し、

（３）　図形グループ管理表を作り、

（４）　サーバプロセスを起動し、

（５）　先に示した図形グループ管理表に従い、サーバプロセスに翻訳処理を分け与え、

（６）　サーバプロセスが生成した画素を受信し、印刷を実施する、

これに対しサーバプロセスは

（１）　中間コードの段階まで翻訳された図形グループを受信し、

（２）　画素に翻訳し、クライアントプロセスに返送する、

という処理を行なう。

コンピュータ6a、6b、6cのいずれかにおいてアプリケーションソフトウェアが発生した印刷要求が、印刷装置159cを対象として行なわれたとすると、印刷装置159cは分散処理によって印刷を行なうために、PDLの翻訳を分担し実行するためのサーバプロセスを必要とする。このために、

手順1:印刷装置159aに搭載された翻訳ソフトウェア160が、この印刷装置159aのメモリ及びプロセッサ資源を使用してクライアントプロセスを作る。

手順2:手順１で作られたクライアントプロセスは、ページ記述言語ソースコードから印刷画素を生成する処理を分散処理するためのタスク生成が可能か問い合わせるメッセージをネットワーク上の他の情報機器に転送する。

手順3:手順２のメッセージに応答する可能性のあるのは、図35の構成ではソフトウェア160が実装されている印刷装置159a、159bだけであるから、その一方又は両方でサーバプロセスが作られる。

手順4:手順３で生成したサーバプロセスを用いて分散処理を行なう。

具体的な上記の個々の手順は実施例１から３までに等しい。

次に、第５の実施例の図36を参照して説明する。

本実施例は第４の実施例を変形したものである。図36において、印刷装置161a, 161b, 161cだけが専用のネットワーク162で相互接続されている。各印刷装置161a, 161b, 161cと各コンピュータ163a, 163b, 16cは、それぞれインターフェース164a, 164b, 164cにより１対１で接続されている。インターフェース164a, 164b, 164cは、この場合、単方向にのみデータ転送が行なわれ得るような例えばセントロニクス準拠等の仕様のもので良い。動作については図35の構成のそれと同様であるが、図36では印刷装置とコンピュータの接続が従来の構成と変わらない等の点で利点がある。

次に第６の実施例を説明する。

PDLを用いた印刷処理の普及に伴い、PDLも複数の仕様に基づき、幾つかの種類を持つに至っている。このため、多くのアプリケーションを使用する場合、複数種類のPDLを印刷処理する必要性が生じる。そこで、既に何

回か述べたラスタライザを、より一般化して使用することが考えられる。

図37は、本発明の第６の実施例の構成図である。ここでは、３台のコンピュータ6a、6b、6cがネットワーク７に接続され使用されている。またプリントコンフィグレーションサーバ401が、ネットワーク７に接続されている。さらに、プリントコンフィグレーションサーバ401と、印刷装置21a、21b、21cとが、ネットワーク403により相互接続されている。ここで、ネットワーク403および７は、それぞれ独立したネットワークである。

本実施例の処理の概要は次の通りである。

まず使用者が、印刷要求を、マウス操作等により行なうと、イベント処理プロセス203により要求が受け取られ、現在使用中のアプリケーション18の印刷処理ルーチンにより、PDLソースコードにより記述された出力ファイル19が生成される。プリントプロセス402は、これをファイルシステムにより読み込み、プリントコンフィグレーションサーバ401に転送する。コンピュータ6aの処理は、この段階で完了である。他の実施例と異なり、コンピュータ6b、6c等も分散処理を行なうことはない。本実施例においては、プリントコンフィグレーションサーバ401上のプロセス406が、これ以降の処理の分散と、並列実行等の同期を管理する。

以下、翻訳印刷処理について、その処理の流れを説明する。

プロセス402によりネットワーク７に出力された印刷データは、PDLのソースコードにより記述されたソースファイル19を、プリントコンフィグレーションサーバ401との接続に必要な形式に翻訳したものである。このフォーマットには、印刷出力をしたいプリンタの識別子、用紙設定、印刷処理を行なったユーザのユーザIDがヘッダとして付加される。ネットワーク403には、複数台のプリンタが接続されているため、印刷処理要求の有ったコンピュータ6aの使用者は、最も近い場所にある印刷装置を指定したい等の要求を持つはずである。このため、使用者からは、印刷装置のどれかを特定できる必要があり、プロセス402は、これを指定する。上記ヘッダに続いて、使用するPDLのタイプ指定および実際のPDLソースコード列が続く。プリントコンフィグレーションサーバ401は、ネットワークデバイスドライバ213から、上記データ列を受け取り、一次翻訳処理を行なう。ここでまずPDLソースコードから、デバイス座標系に写像された（すなわち印刷装置の解像度に依存した形式の）記述による中間コードが得られる。続いてこれら中間コードは、既に幾つかの実施例で述べたように、並列実行可能な小部分に分割され、ネットワーク403を通じて印刷装置21a、21b、21cに転送される。これら中間コードを受け取った印刷装置21a、21b、21cでは、インターフェースドライバ303が、パケットを取り出し、データ列を解析処理プロセス405に渡す。

各印刷装置21a、21b、21cでの、処理プロセス群の構成を図38に示す。

各印刷装置21a、21b、21cでは、ラスタライザ404が画素発生を行なうが、他の実施例で示したラスタライザ212が特定のページ記述言語を前提とした構成であったのに比較し、この実施例のラスタライザ404は、グラフィクスプリミティブインターフェース407を持つ点が異なっている。

印刷装置では、画素はイメージメモリ上でも、実画素としても点列に量子化されている。このため、曲線データは、実際は微小な線分からなる折れ線の連続である。ラスタライザ404の処理の主たる部分は、直線発生処理408及び点列発生処理409によって占められる。

各印刷装置21a、21b、21cにおいて、受け取られた中間コードは、解析処理プロセス405により、ページ記述言語仕様に依存しない、直線曲線の集合に変換される。これら変換結果を受け取ったグラフィクスプリミティブインターフェース407は、それぞれの曲線に合わせ、折れ線近似を行ない、直線発生処理408に渡す。これらデータは、点列発生処理409が行なわれ、記述上閉じた領域に対する塗りつぶしが要求された場合は、さらに塗りつぶし処理410が行なわれる。以上の４処理により、ラスタライザ404から取り出されるデータは、画素データに変換済みである。

再び図37を参照して、プリントコンフィグレーションサーバ401のプロセス406では、各印刷装置21a、21b、21cのラスタライザ404からのデータを受信し、イメージメモリ上で合成し、印刷処理を行なう印刷装置、例えば、21aに転送する。前述のように、プロセス402がサーバ401に対し転送したデータ列には、印刷処理を行ないたい印刷装置の識別子が記述されている。従って、ここで印刷出力を行なうのは、識別子の指定する印刷装置21aである。このため、印刷装置21a、21b、21cはサーバ401から見たとき、個々に識別可能なネットワークアドレス機構を持つ必要がある。本実施例のネットワーク403は、物理的なアドレス情報を持つ印刷装置プリンタ21a、21b、21cに対し、ネットワークアドレスと、プリンタの固有名称を結び付けたアドレス管理を行なう。前述の実施営と同様に、これはアドレス管理表により、サーバ401上のプログラムにより管理される。

以上の様に、プリントコンフィグレーションサーバ401を導入し、プリンタ側においてラスタライズのみを並列処理する構成が実現できる。この構成を用いた場合、ページ記述言語ソースコードから、グラフィクスプリミティブへの翻訳をサーバ401上のプロセス406が行なうことにより、印刷装置21a、21b、21cのラスタライザ404は、ページ記述言語の仕様とは独立して設計可能である。また、サーバ上のプロセス406の翻訳により、複数の仕様のページ記述言語を扱うことが可能となる。

前述した第３の実施例は、明らかに第２の実施例の中

で印刷装置108が果たす役割の一部を装置133に行なわせるものである。また第２の実施例は第１の実施例の内、翻訳系の一部の機能を印刷装置側に移植したものである。

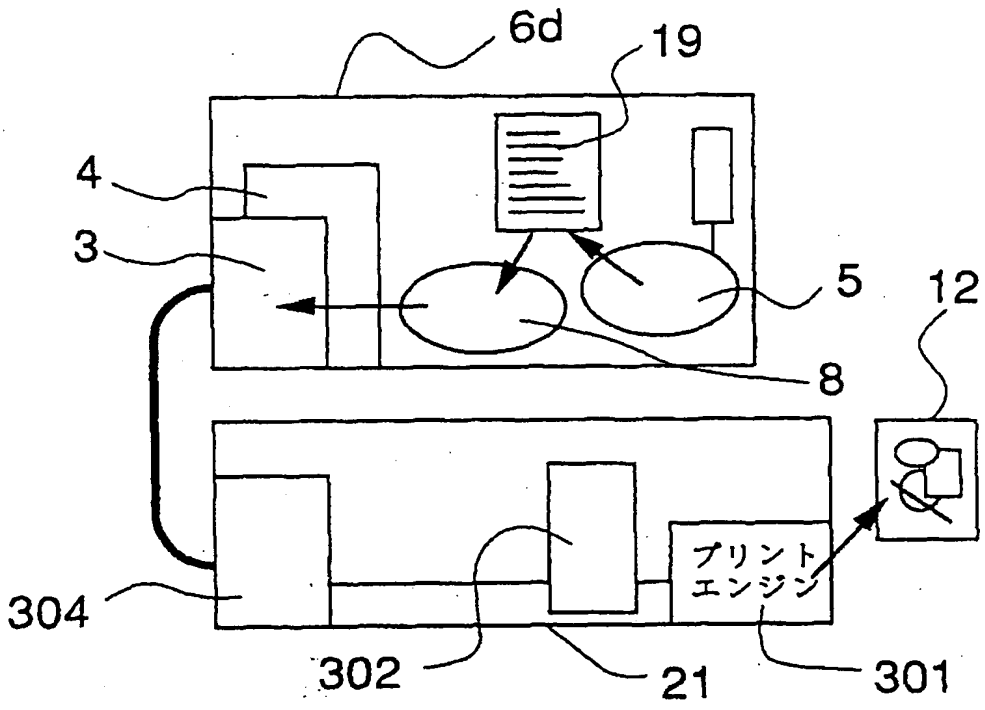
以上の実施例によって示されるように、本発明はPDLにより記述されたファイルを分割し、複数の情報機器に分散し翻訳させるためのものであり、これを実現するソフトウェアの各部分が実際にいかなる情報処理機器のハードウェアの上で実行されるかは本発明において本質的ではない。何らかの双方向通信手段によって結合された複数の情報処理機器において、それがどの様な情報処理機器であるにせよ、本発明が実現できることは、既に述べた幾つかの実施例から理解されるであろう。また、本発明は既に述べた実施例のみに限定されるものではなく、他の種々の態様で実施し得ることも明らかである。

現在一般的なオフィス環境にあっては、５から20台程度のパーソナルコンピュータに対し、その半数以上の台数のプリンタが設置されている。このような環境で、ある時間にプリンタの稼働状況を見てみると、全てのプリンタが動作中であるということは極めてまれであり、実際印刷処理を行なっているのは数台に過ぎないのが通常である。しかしながらプリンタにはプロセッサもメモリも搭載されており、しかも年々高速大容量に成りつつある。本発明の分散処理インタープリタをこのようなプリンタに搭載し、それらをネットワークによって結び使用すれば、個々のプリンタのパフォーマンスを大幅に上げることが可能となるばかりか、個々のプリンタの性能はある程度低くても構わないため、１台あたりのコストを押さえた製品を使用することも可能になる。これはCPU資源、メモリ資源の利用という点でも効率的である。もちろんコンピュータ上にあって本発明の分散処理インタープリタを使用することも同じ意味で効果がある。

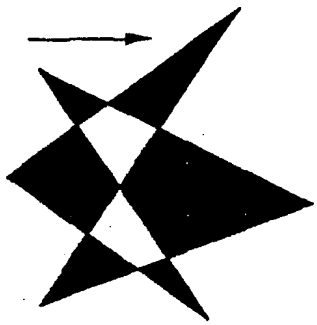
使い将来普及が予想されるカラープリンタを用いたデスクトップパブリッシングにおいては、カラー画像の処理のために必要となるメモリが白黒画像のそれに比較しいかに大きいかを考えると、本発明の有利さは明らかである。



【第１Ａ図】

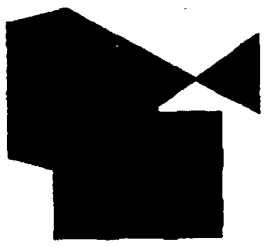



【第１２図】

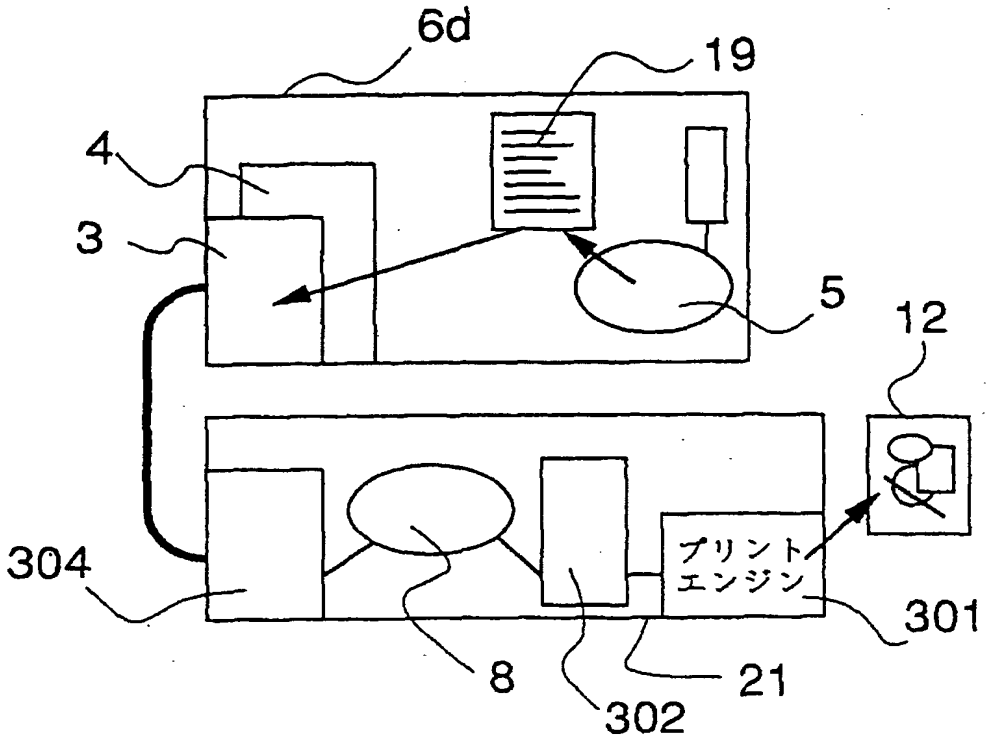


【第１７図】

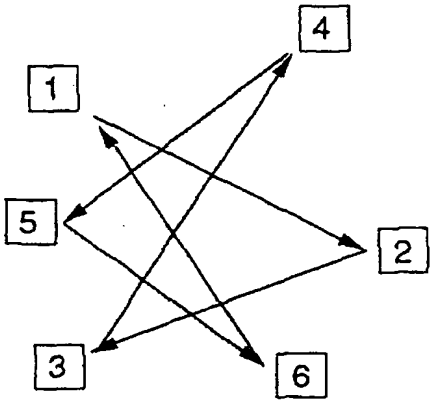
【第１８図】



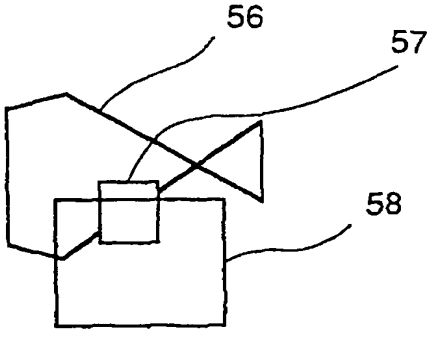
【第１Ｂ図】



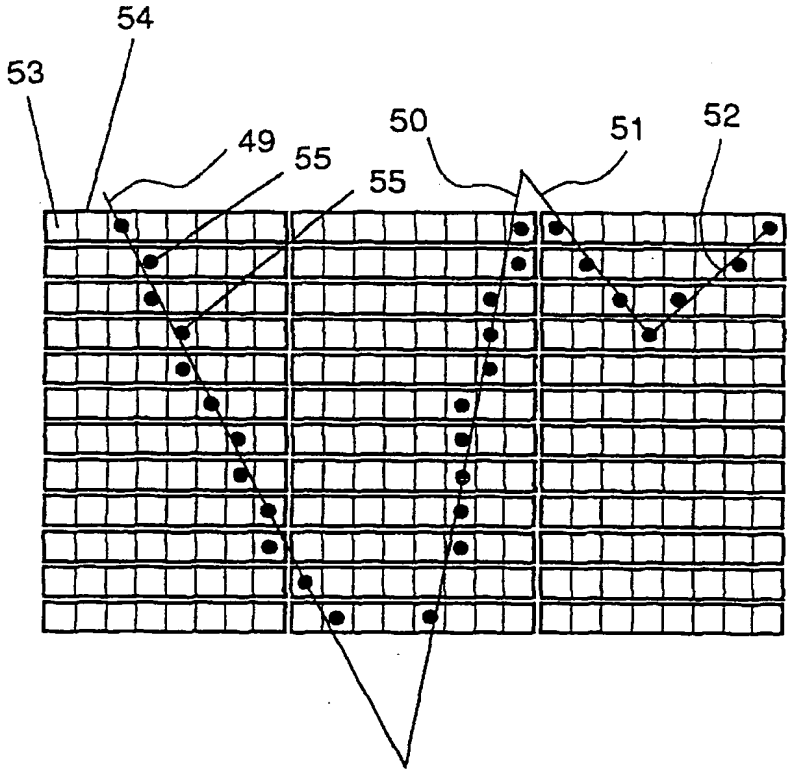

【第１１図】



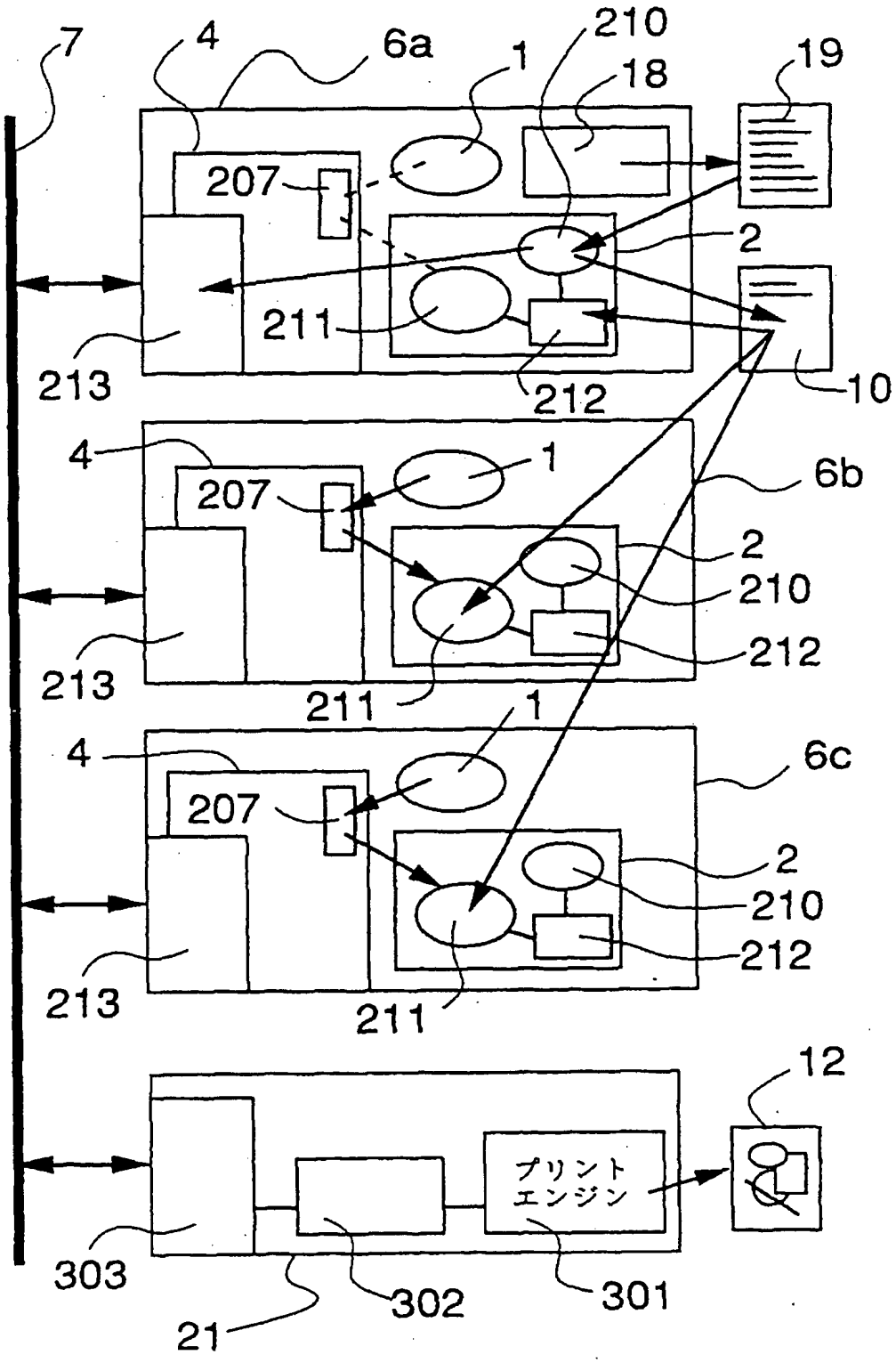

【第１６図】



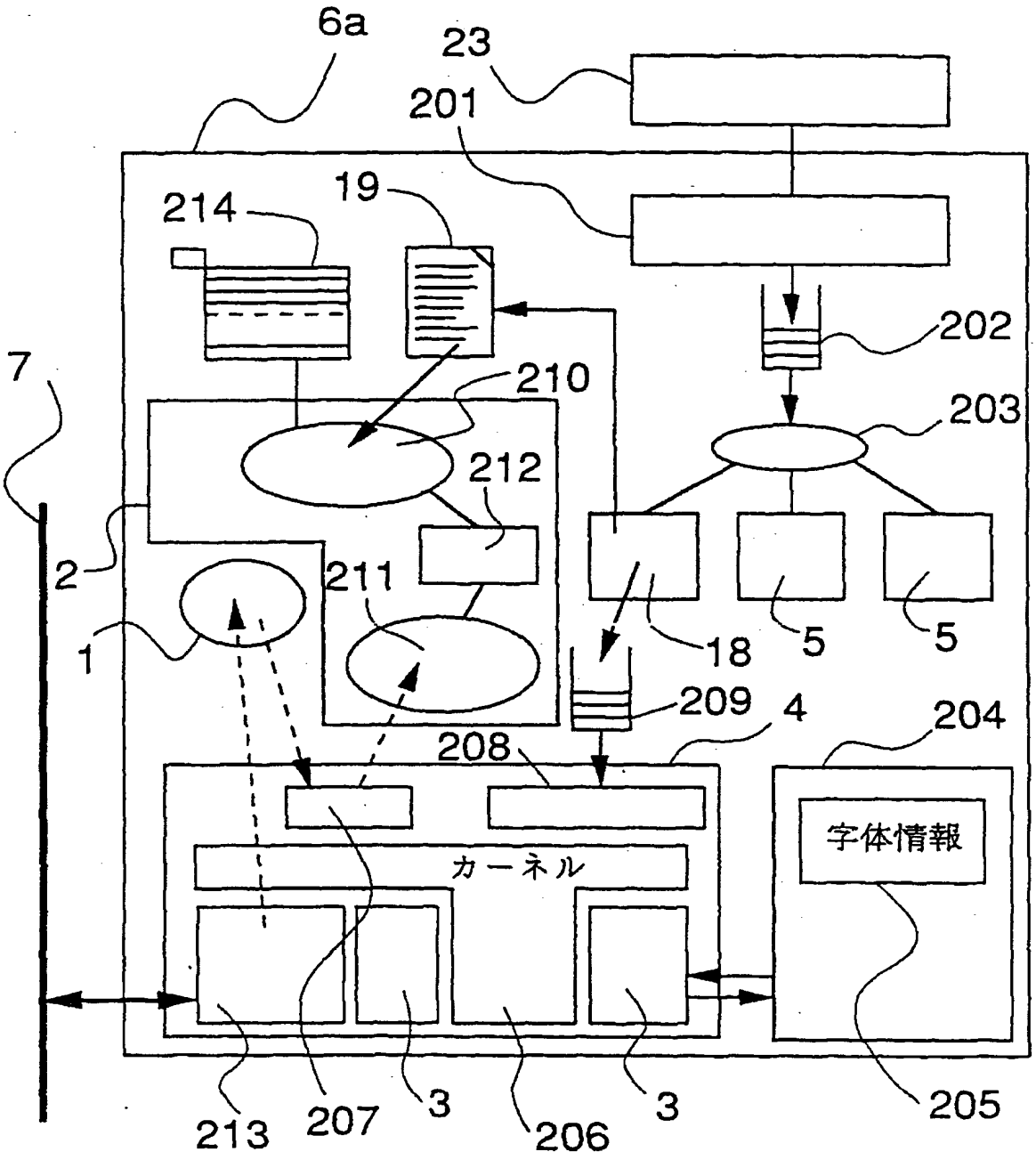

【第１４図】



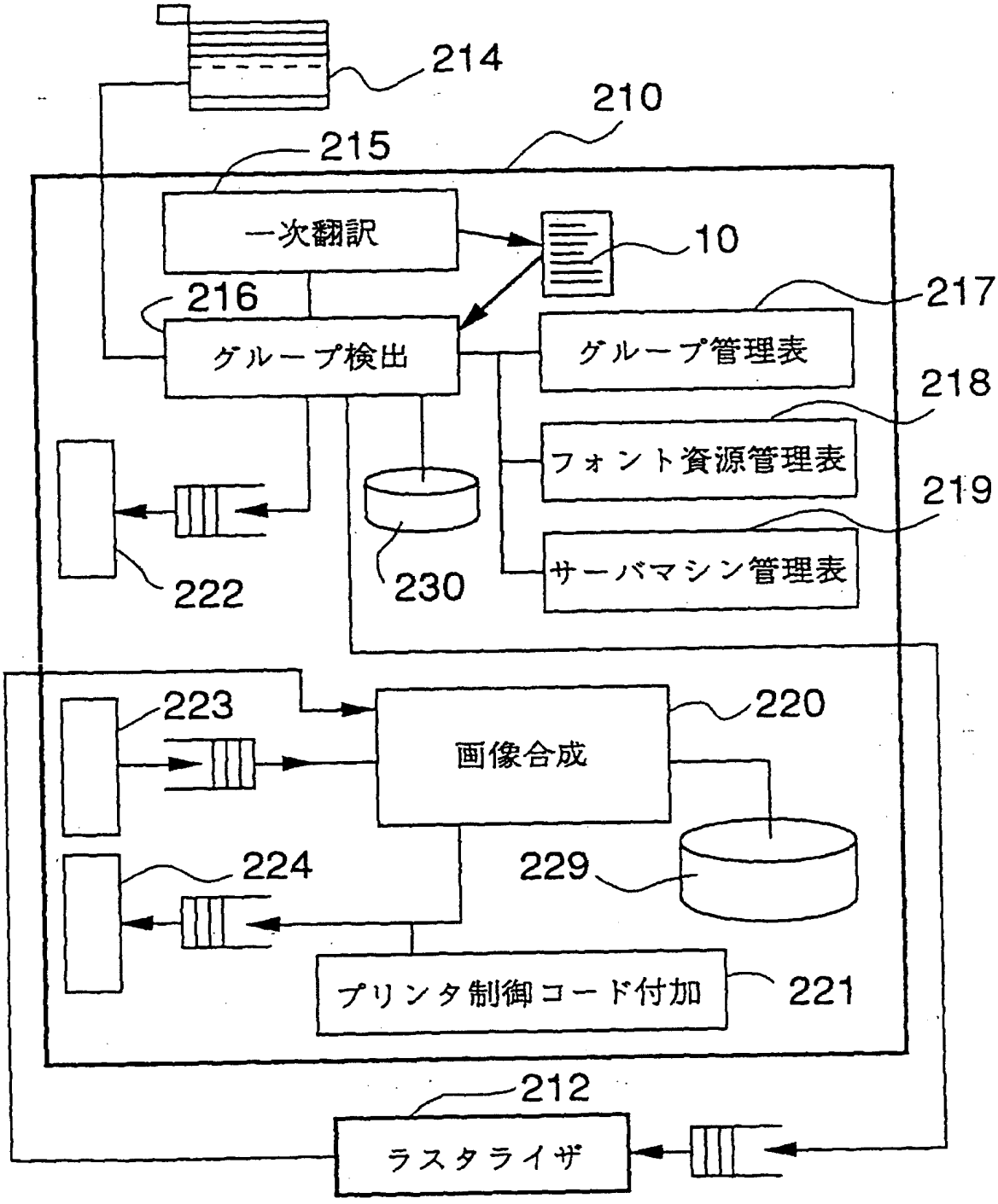
【第２図】


7
4
6a
1
210
18
19
207
2
211
213
212
10
4
1
6b
207
2
210
213
212
211
4
1
6c
207
2
210
213
212
211
12
プリント
エンジン
303
302
301
21

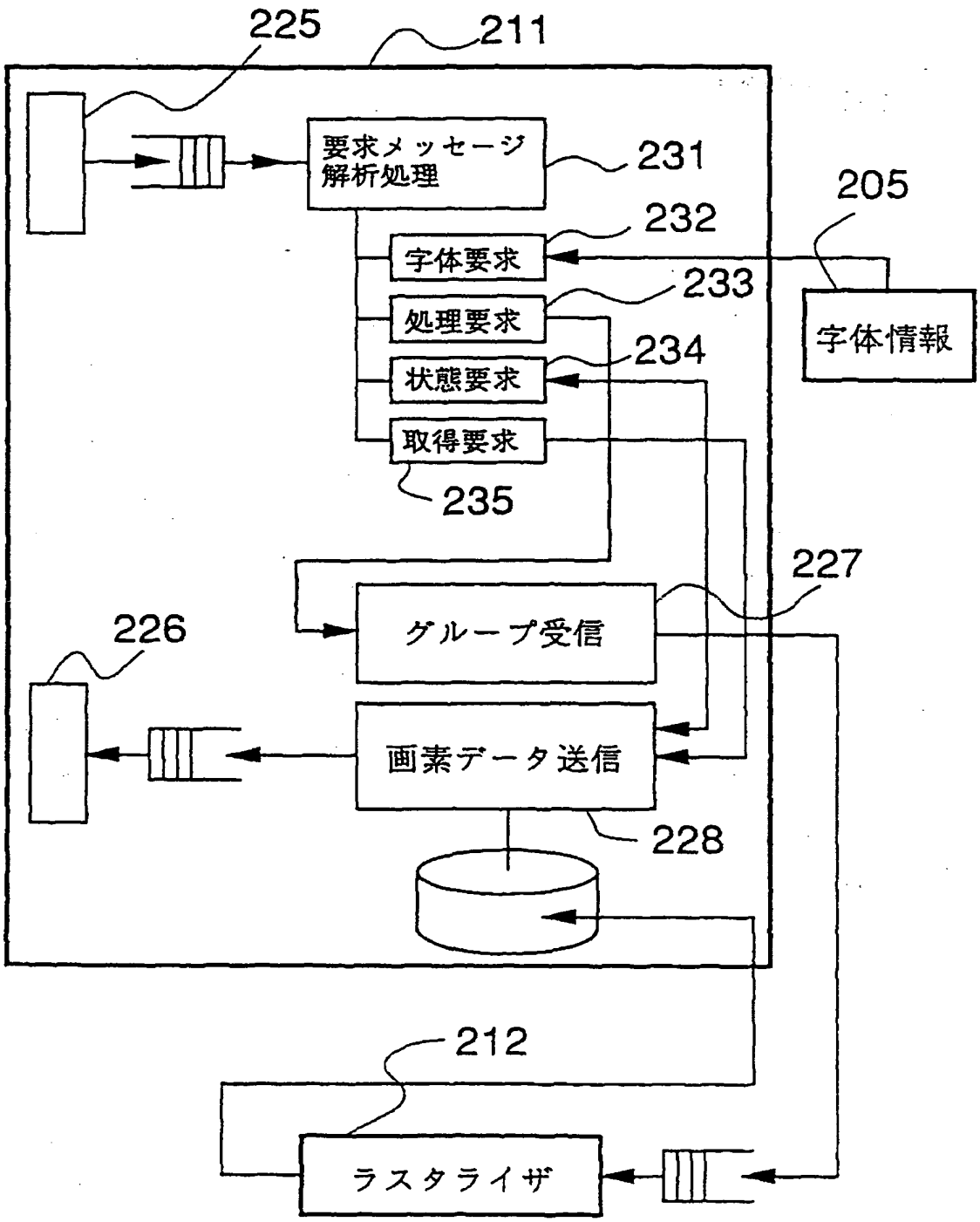
【第３図】


6a
23
201
214
19
202
7
210
203
212
2
211
1
18
5
5
209
4
204
208
字体情報
カーネル
213
207
3
206
3
205

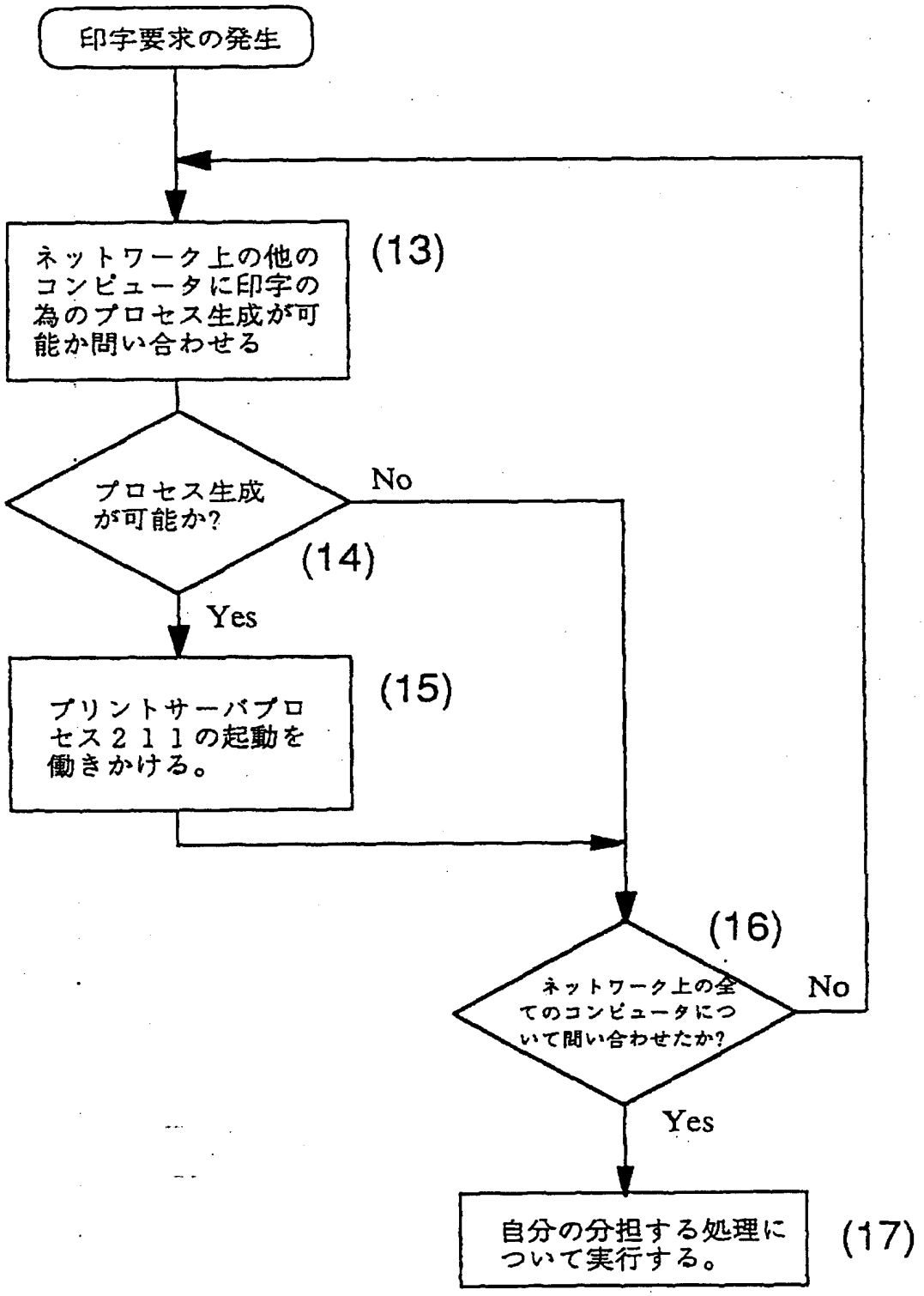
【第４図】


214
210
215
一次翻訳
10
216
217
グループ検出
グループ管理表
218
フォント資源管理表
219
サーバマシン管理表
222
230
223
220
画像合成
224
229
プリンタ制御コード付加
221
212
ラスタライザ

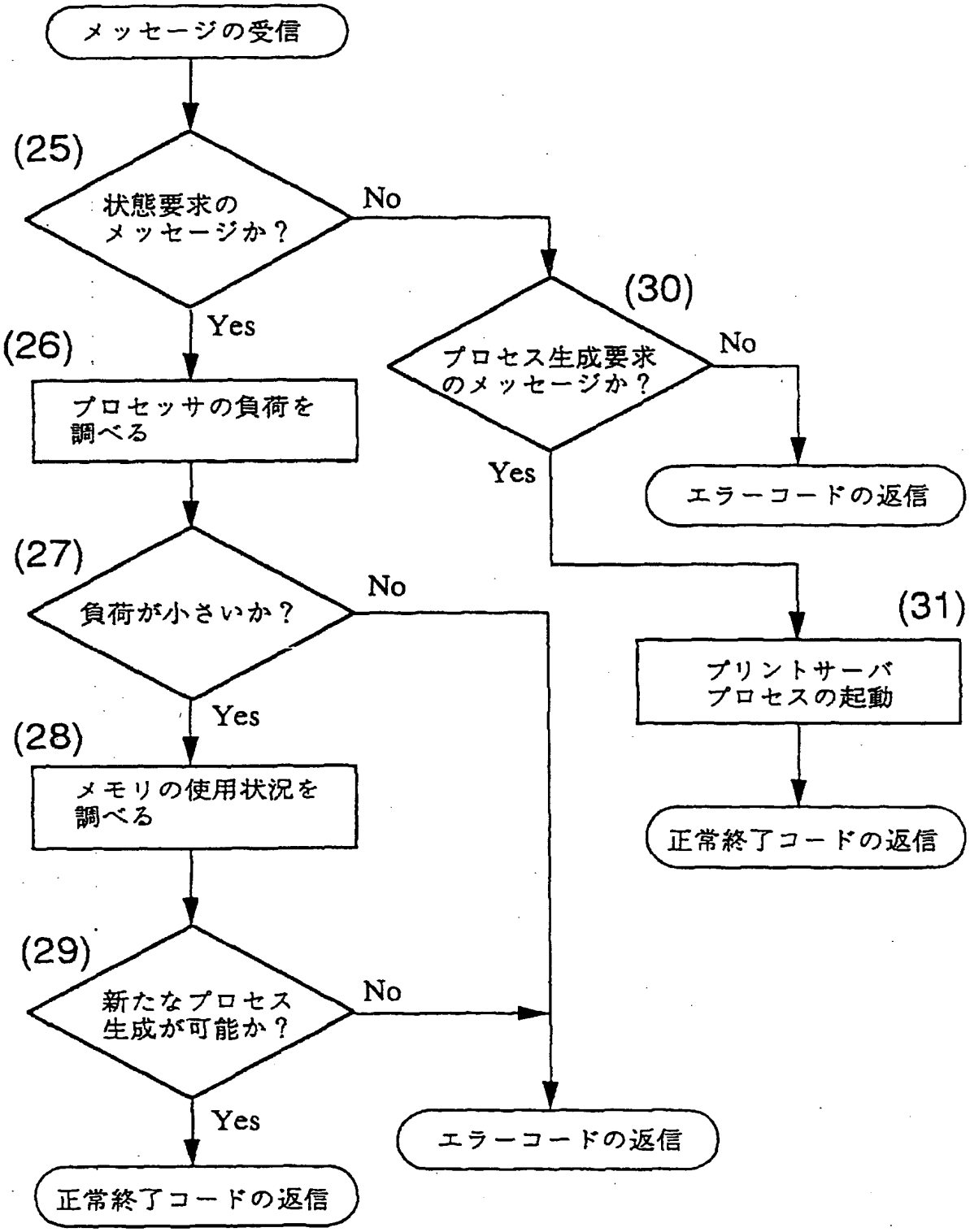
【第５図】


225
211
要求メッセージ
解析処理
231
205
232
字体要求
233
処理要求
字体情報
234
状態要求
取得要求
235
227
グループ受信
226
画素データ送信
228
212
ラスタライザ

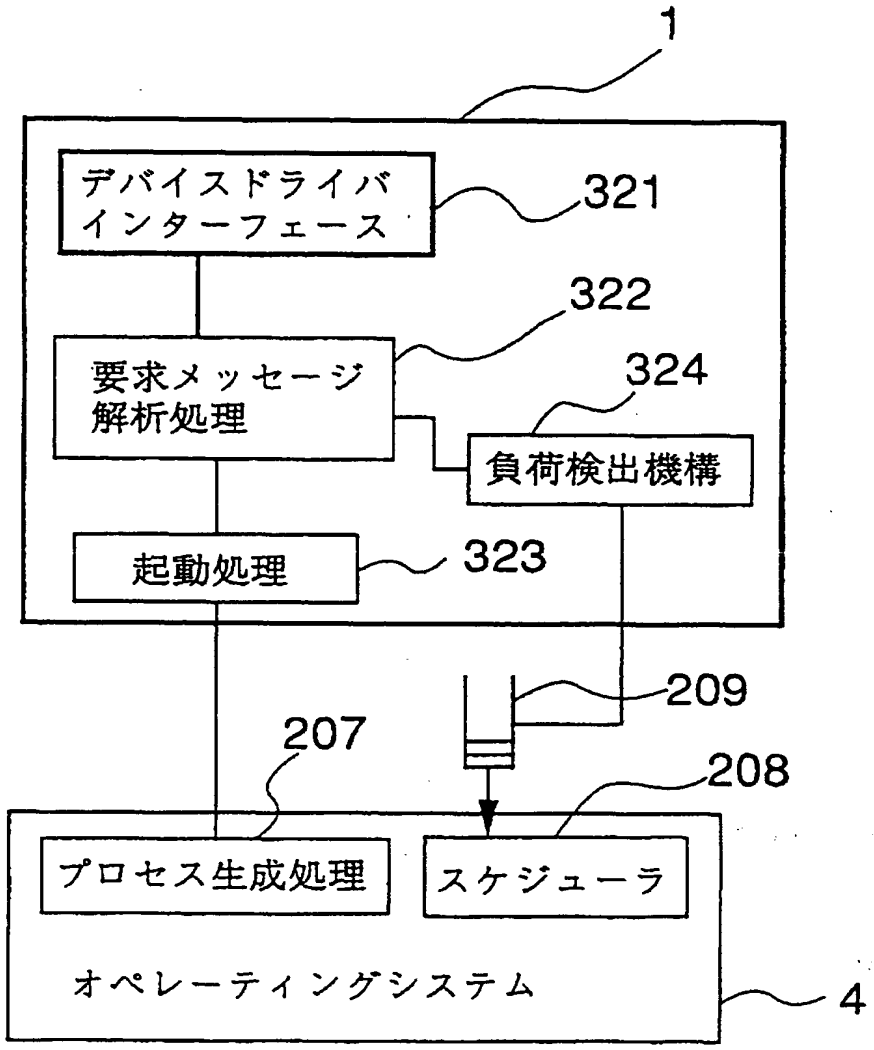
【第６図】


印字要求の発生
ネットワーク上の他のコンピュータに印字の為のプロセス生成が可能か問い合わせる
(13)
プロセス生成が可能か?
(14)
No
Yes
プリントサーバプロセス２１１の起動を働きかける。
(15)
(16)
ネットワーク上の全てのコンピュータについて問い合わせたか?
No
Yes
自分の分担する処理について実行する。
(17)

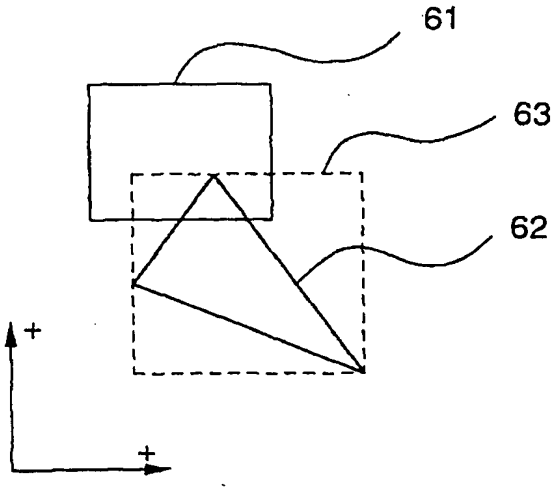
【第７図】


メッセージの受信
(25)
状態要求のメッセージか？
No
Yes
(26)
プロセッサの負荷を調べる
(27)
負荷が小さいか？
No
Yes
(28)
メモリの使用状況を調べる
(29)
新たなプロセス生成が可能か？
No
Yes
正常終了コードの返信
エラーコードの返信
(30)
プロセス生成要求のメッセージか？
No
Yes
エラーコードの返信
(31)
プリントサーバプロセスの起動
正常終了コードの返信

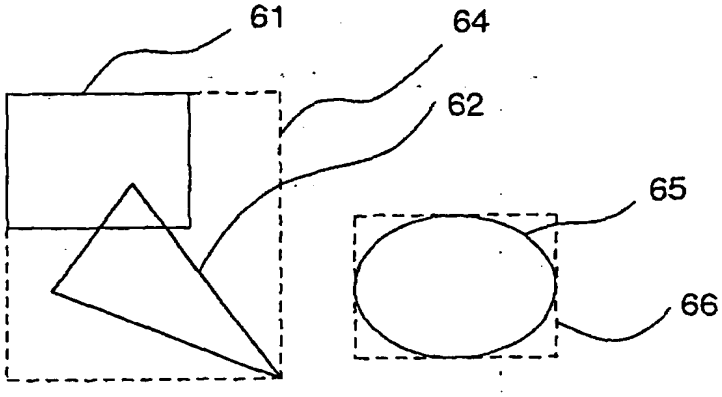
【第８図】



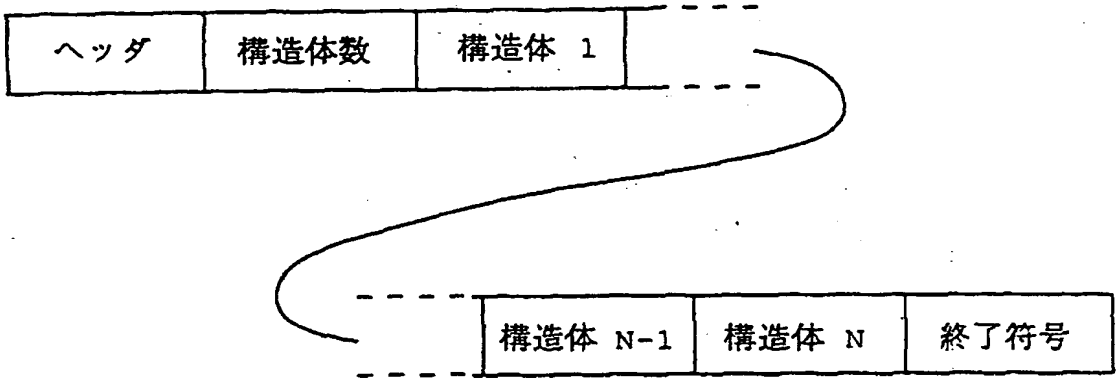

【第１９図】



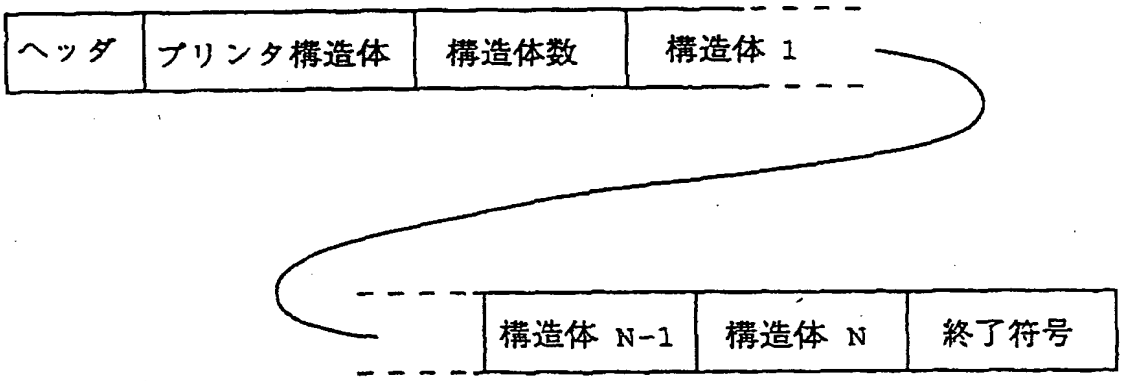

【第２０図】



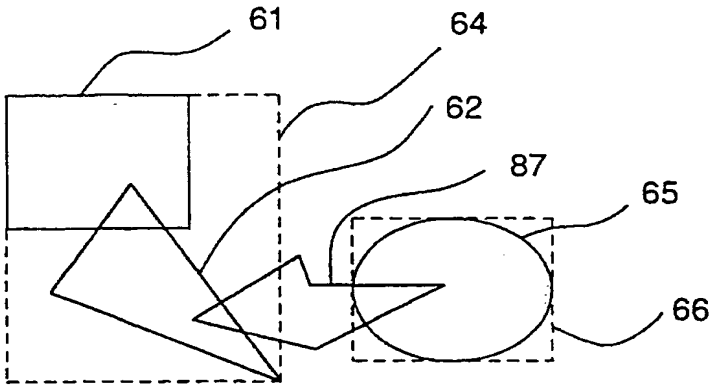
【第９図】




【第１０図】

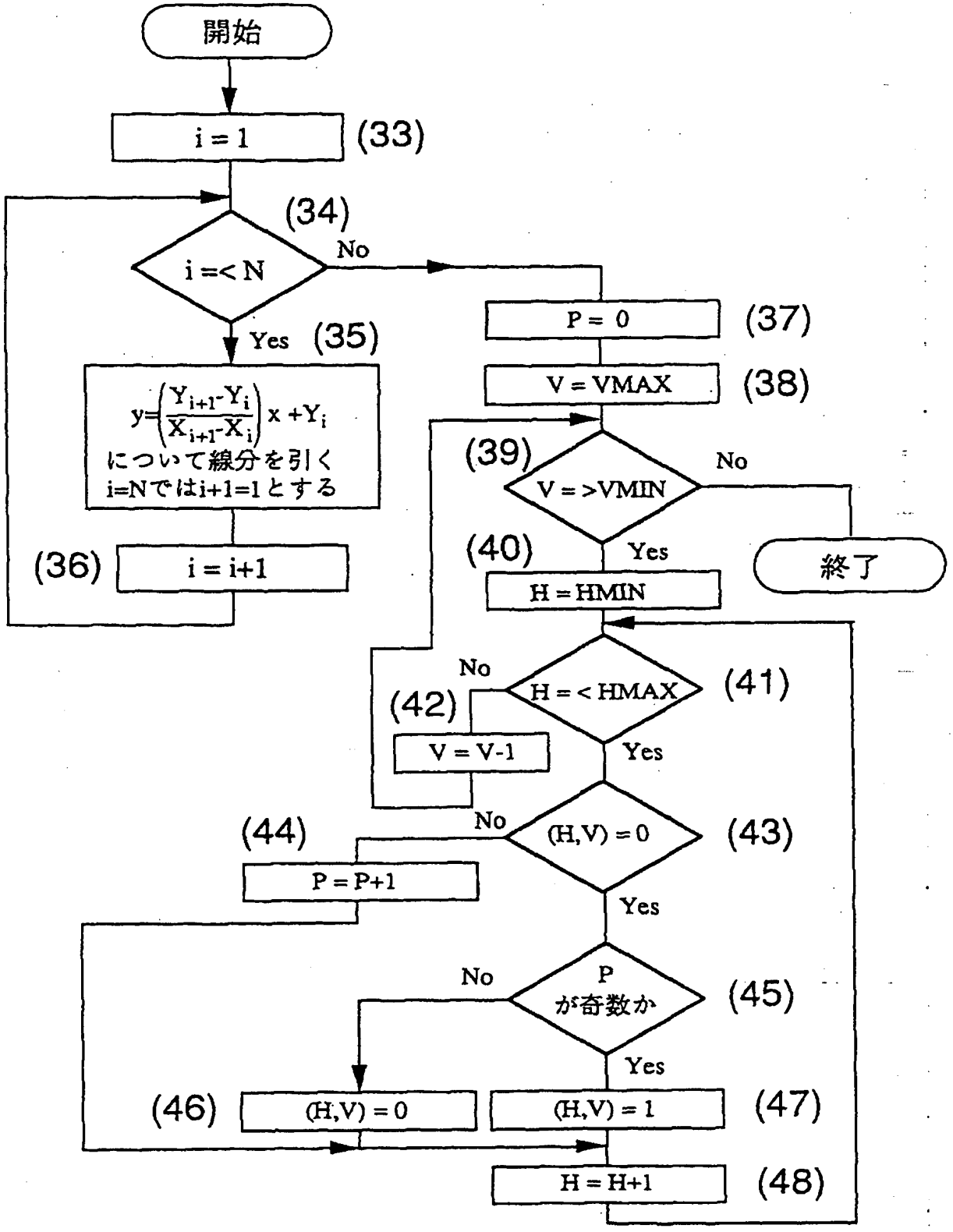



【第２３図】

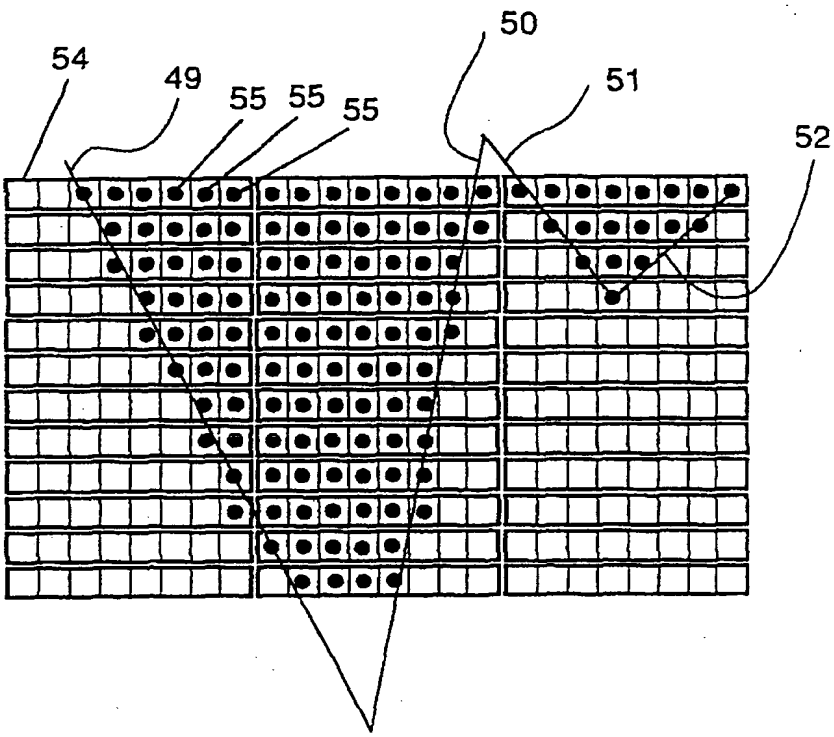


(33)

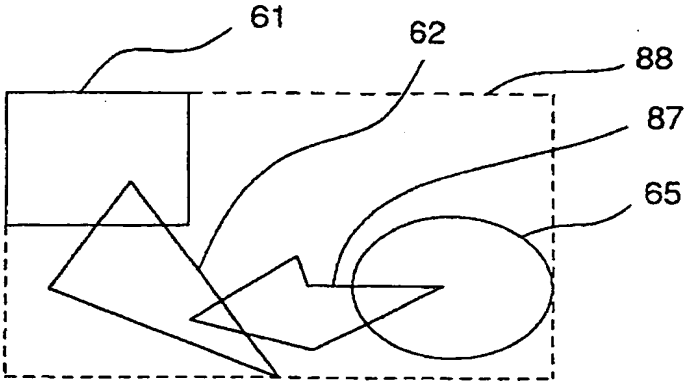
【第１３図】


開始
i = 1
(33)
(34)
i =< N
No
Yes
(35)
$y=\left(\frac{Y_{i+1}-Y_i}{X_{i+1}-X_i}\right)x+Y_i$
について線分を引く
i=Nではi+1=1とする
(36)
i = i+1
P = 0
(37)
V = VMAX
(38)
(39)
V = >VMIN
No
Yes
終了
(40)
H = HMIN
No
(41)
H = < HMAX
(42)
V = V-1
Yes
No
(43)
(H,V) = 0
(44)
P = P+1
Yes
No
P
が奇数か
(45)
Yes
(46)
(H,V) = 0
(H,V) = 1
(47)
H = H+1
(48)

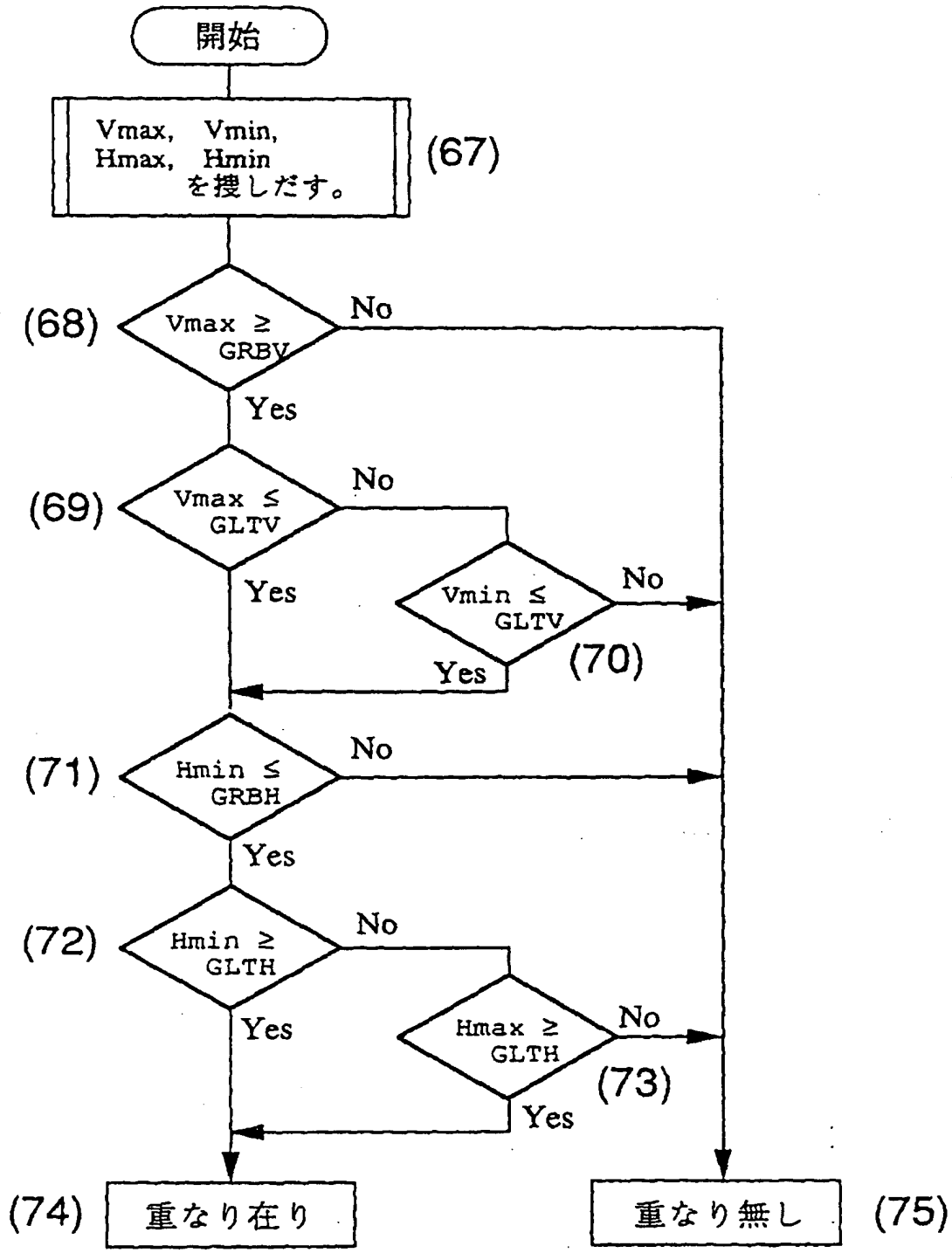
【第１５図】



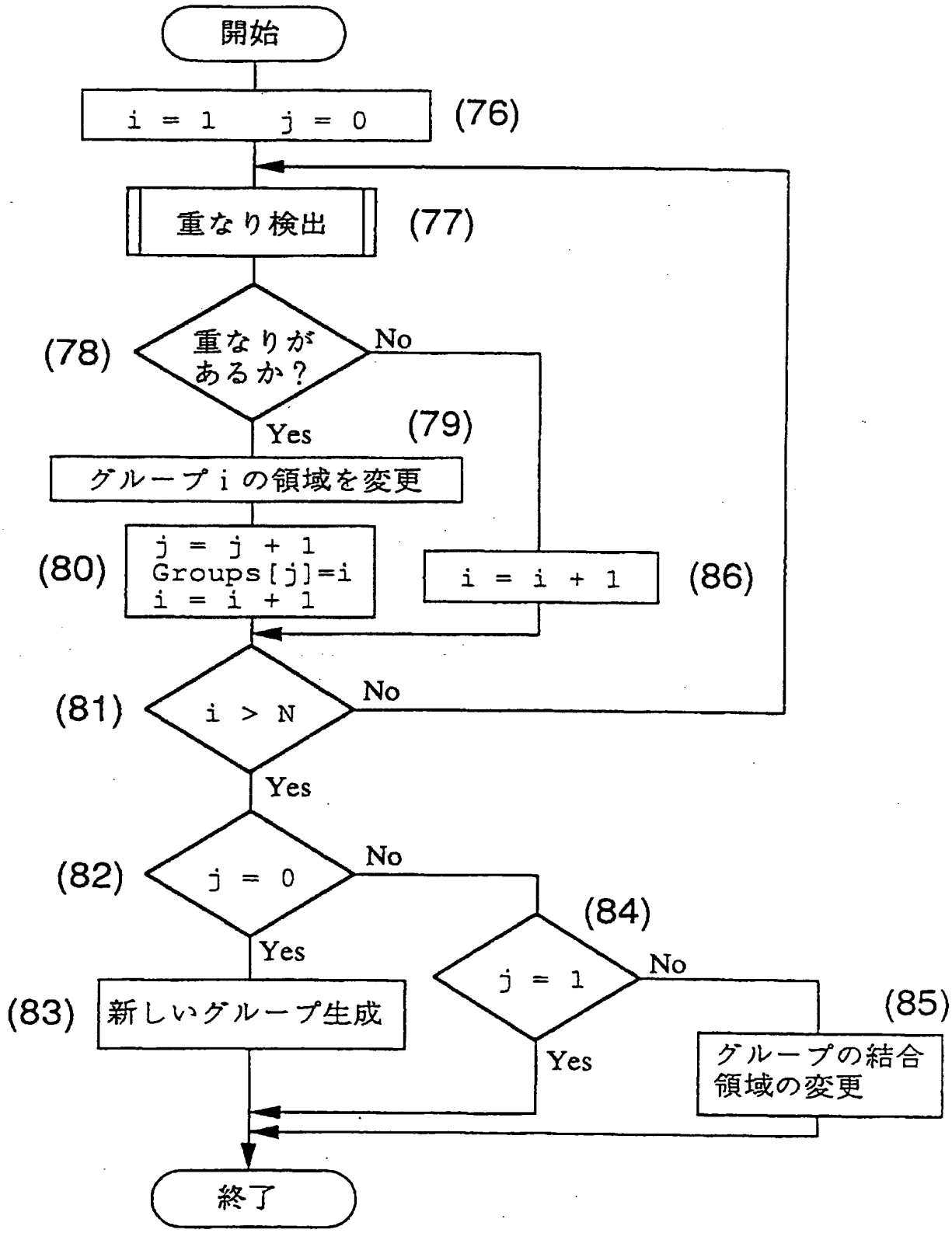

【第２４図】



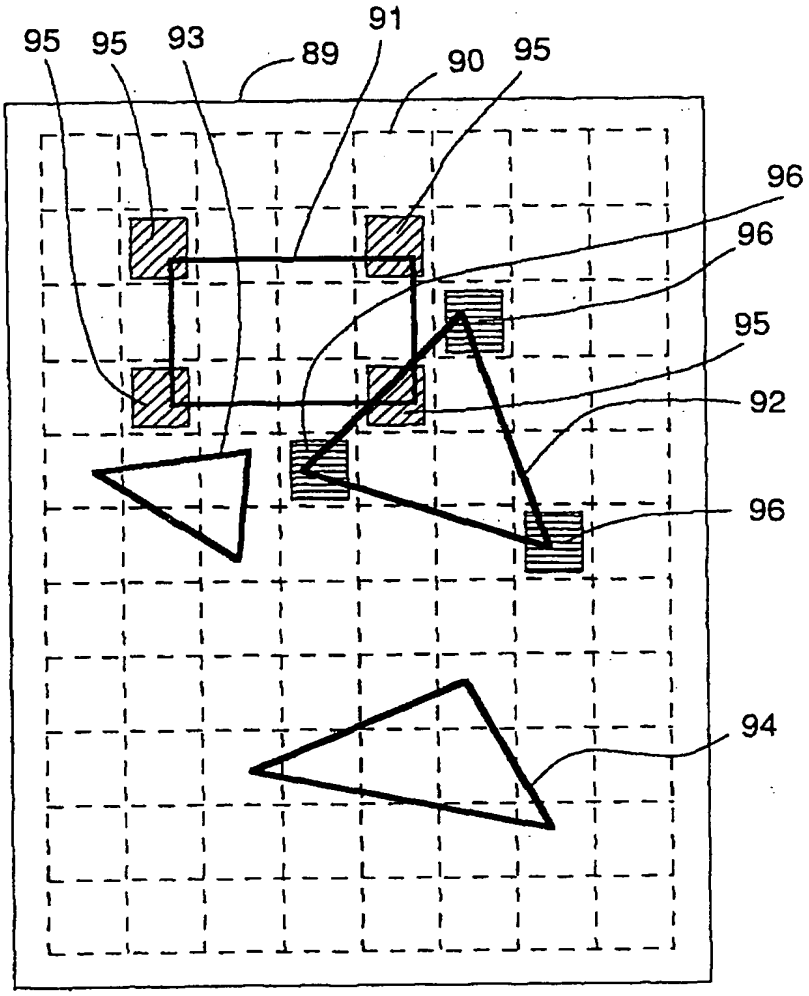
【第２１図】


開始
Vmax, Vmin,
Hmax, Hmin
を捜しだす。
(67)
(68)
Vmax ≥ GRBV
No
Yes
(69)
Vmax ≤ GLTV
No
Yes
Vmin ≤ GLTV
No
(70)
Yes
(71)
Hmin ≤ GRBH
No
Yes
(72)
Hmin ≥ GLTH
No
Yes
Hmax ≥ GLTH
No
(73)
Yes
(74)
重なり在り
重なり無し
(75)

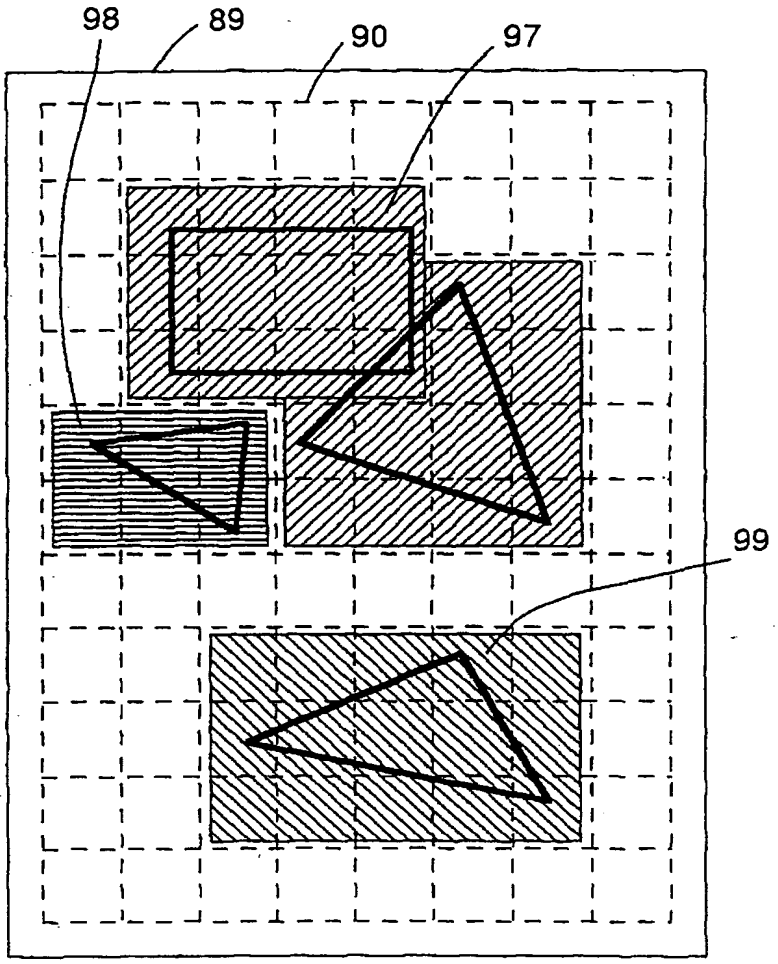
【第２２図】


開始
i = 1　j = 0
(76)
重なり検出
(77)
(78)
重なりがあるか？
No
Yes
(79)
グループｉの領域を変更
(80)
j = j + 1
Groups[j]=i
i = i + 1
i = i + 1
(86)
(81)
i > N
No
Yes
(82)
j = 0
No
Yes
(84)
j = 1
No
Yes
(83)
新しいグループ生成
(85)
グループの結合領域の変更
終了

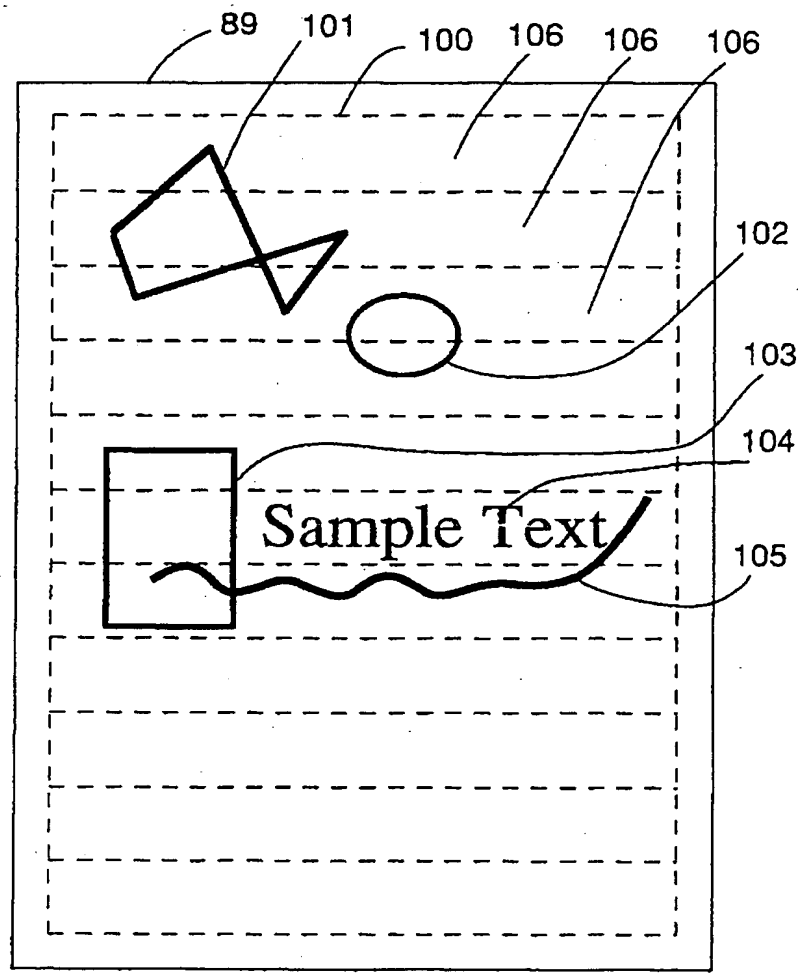
【第２５図】



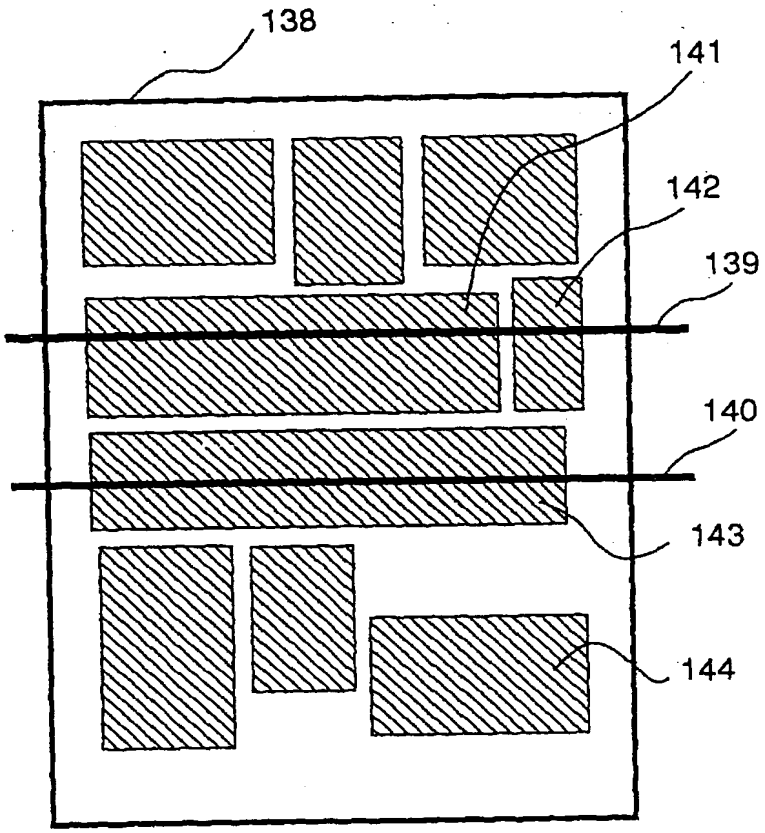
【第２６図】



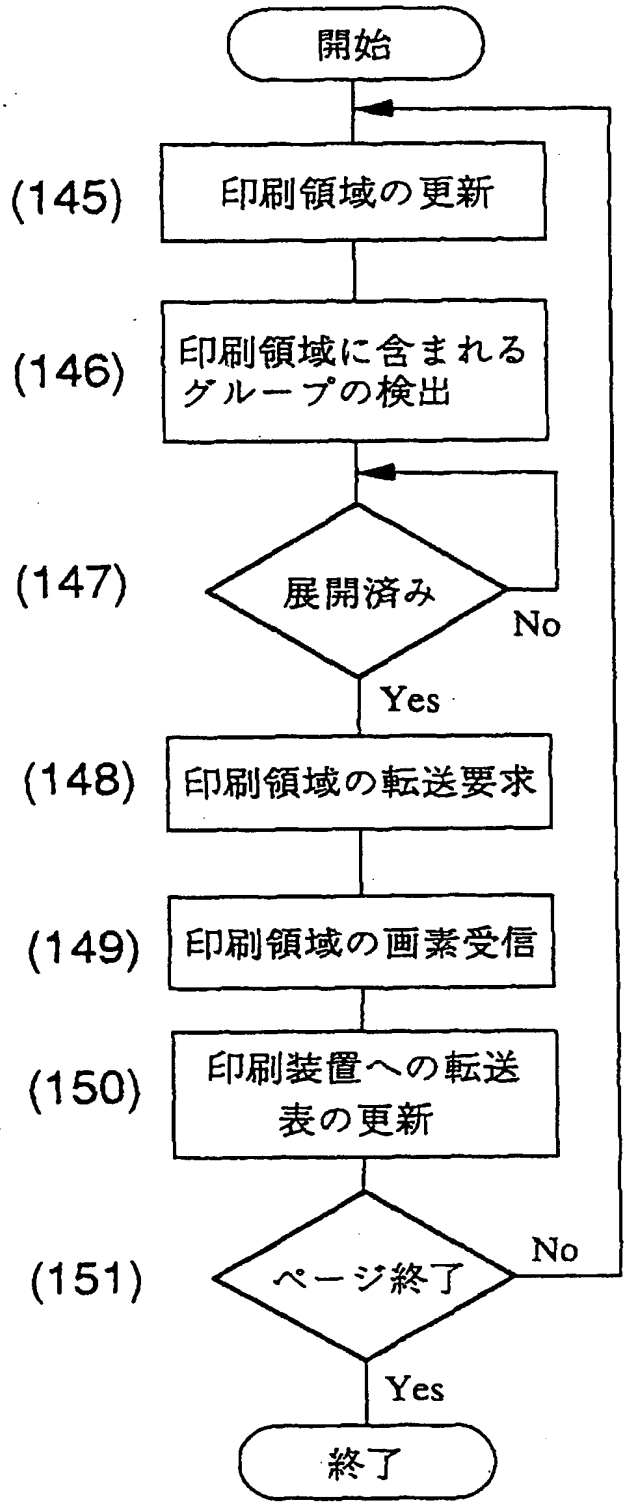
【第２７図】



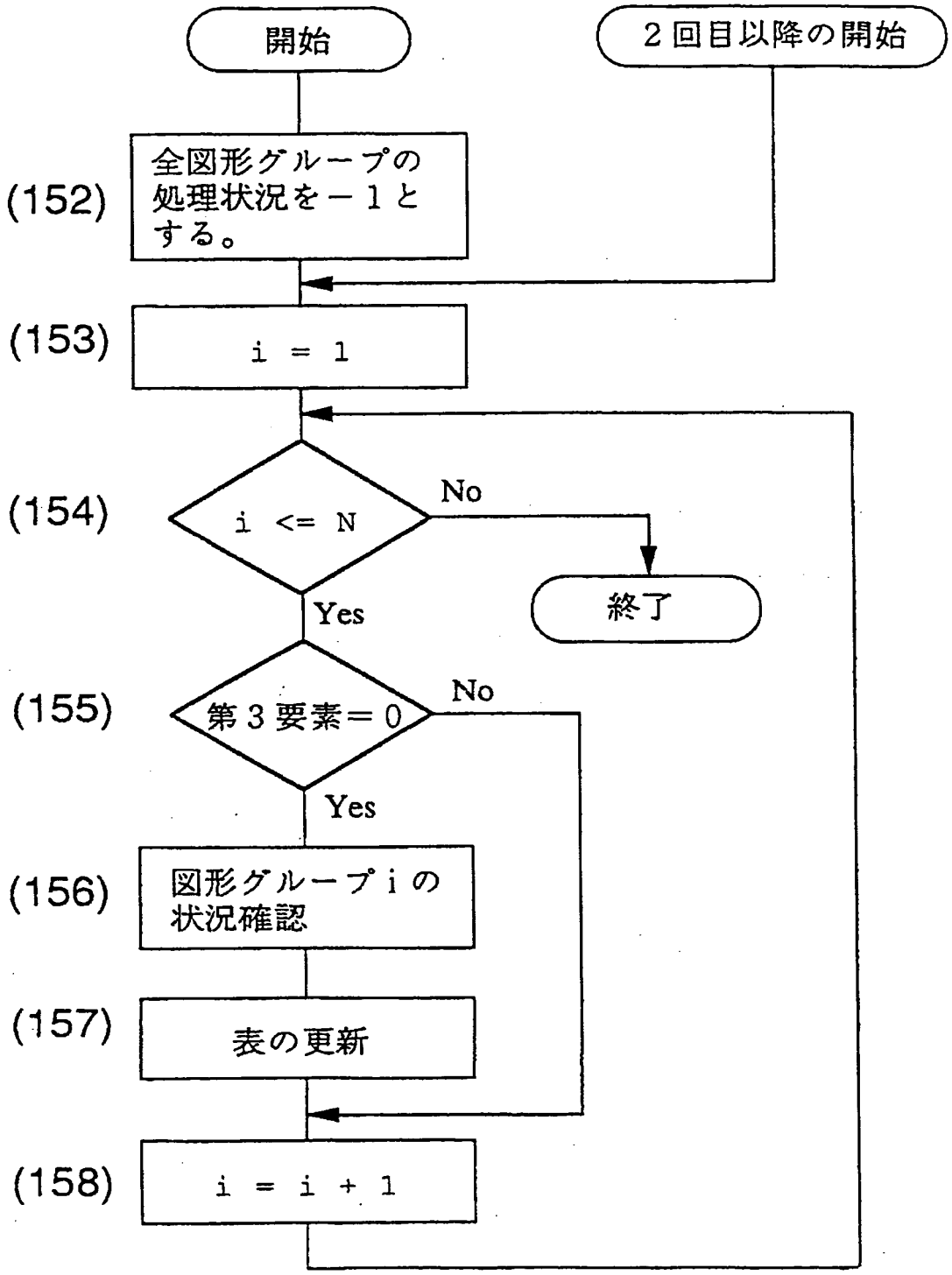
【第２８図】



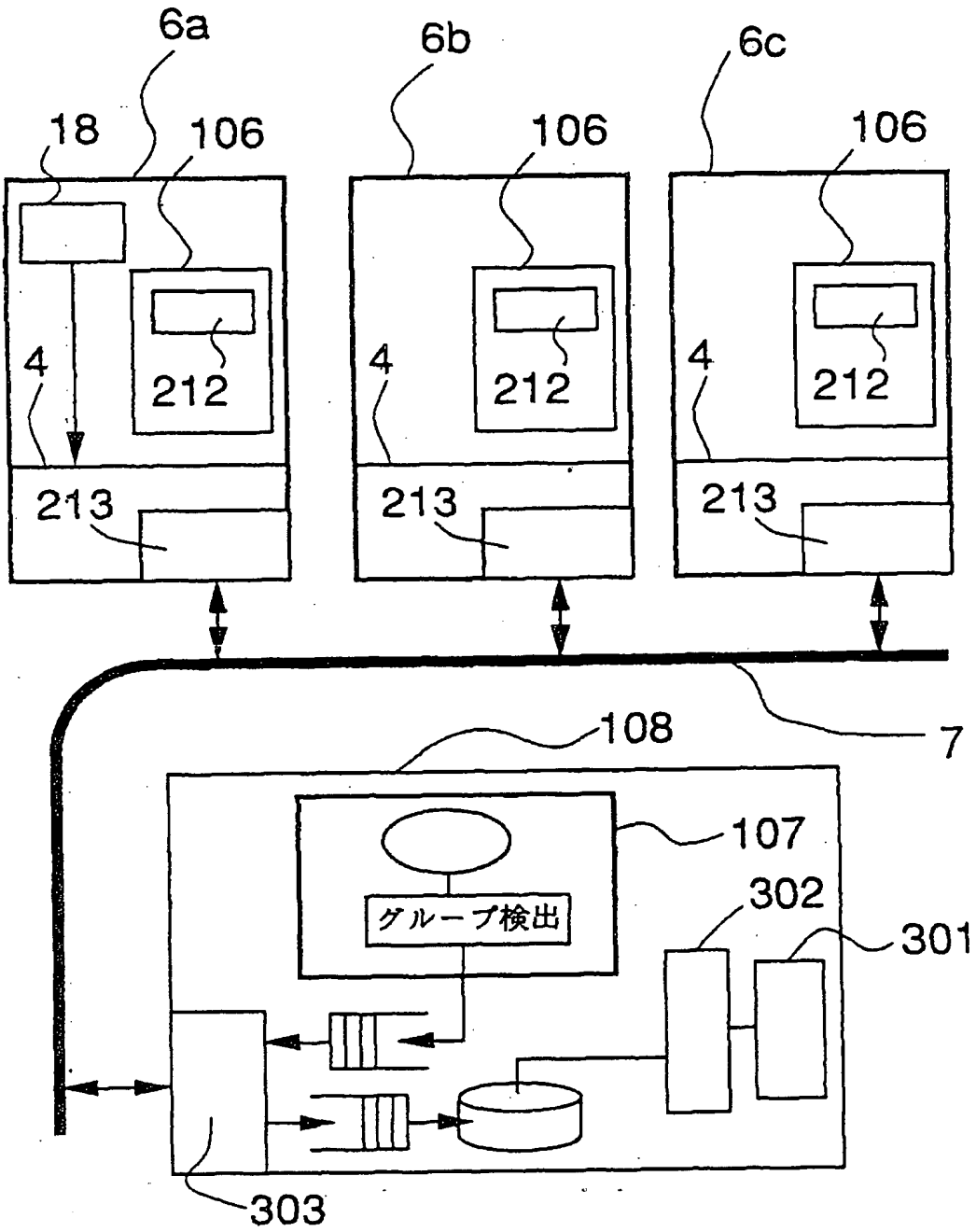
【第２９図】


開始
(145)
印刷領域の更新
(146)
印刷領域に含まれるグループの検出
(147)
展開済み
No
Yes
(148)
印刷領域の転送要求
(149)
印刷領域の画素受信
(150)
印刷装置への転送表の更新
(151)
ページ終了
No
Yes
終了

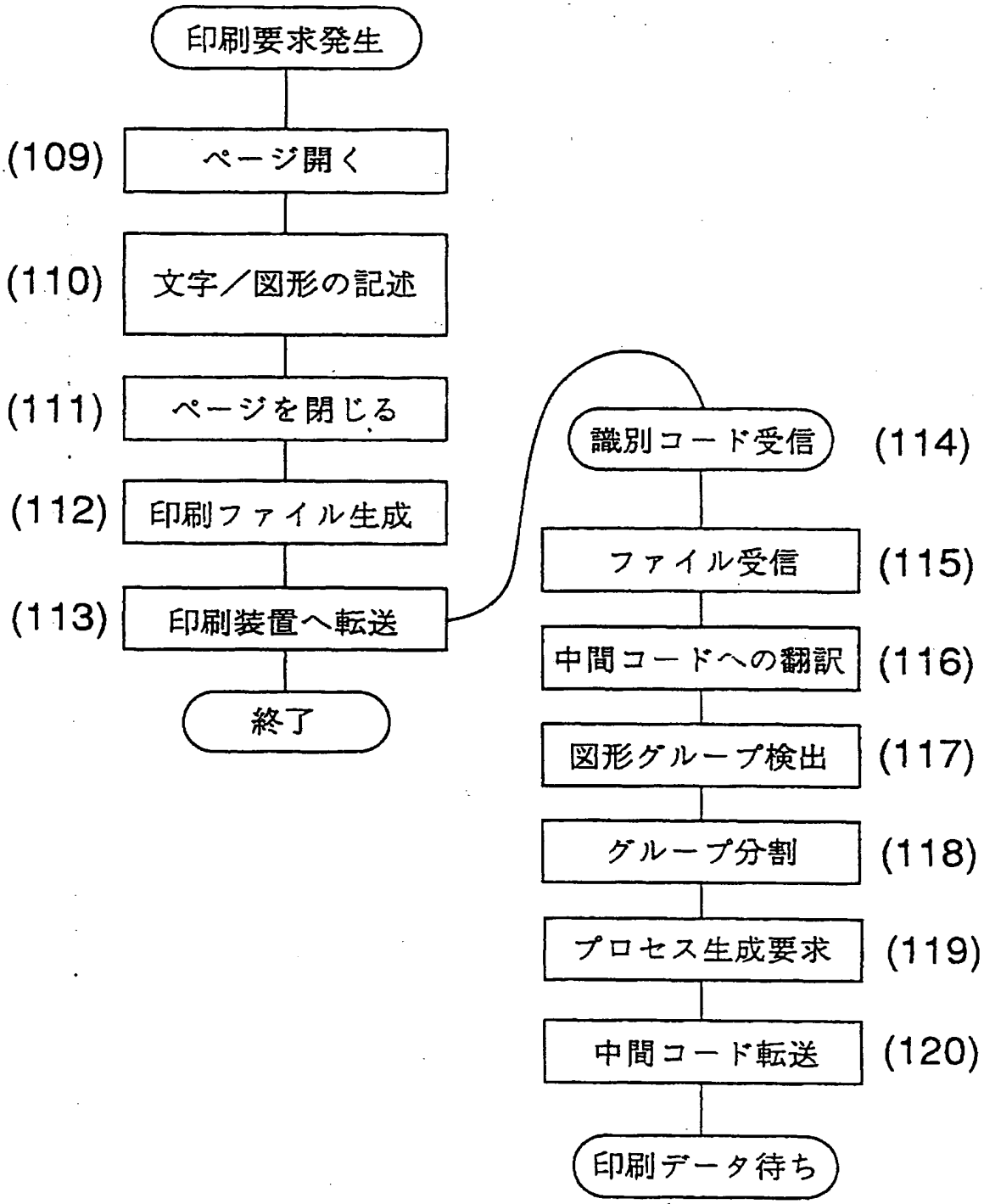
【第３０図】


開始
２回目以降の開始
(152)
全図形グループの処理状況を－１とする。
(153)
i = 1
(154)
i <= N
No
終了
Yes
(155)
第３要素＝０
No
Yes
(156)
図形グループｉの状況確認
(157)
表の更新
(158)
i = i + 1

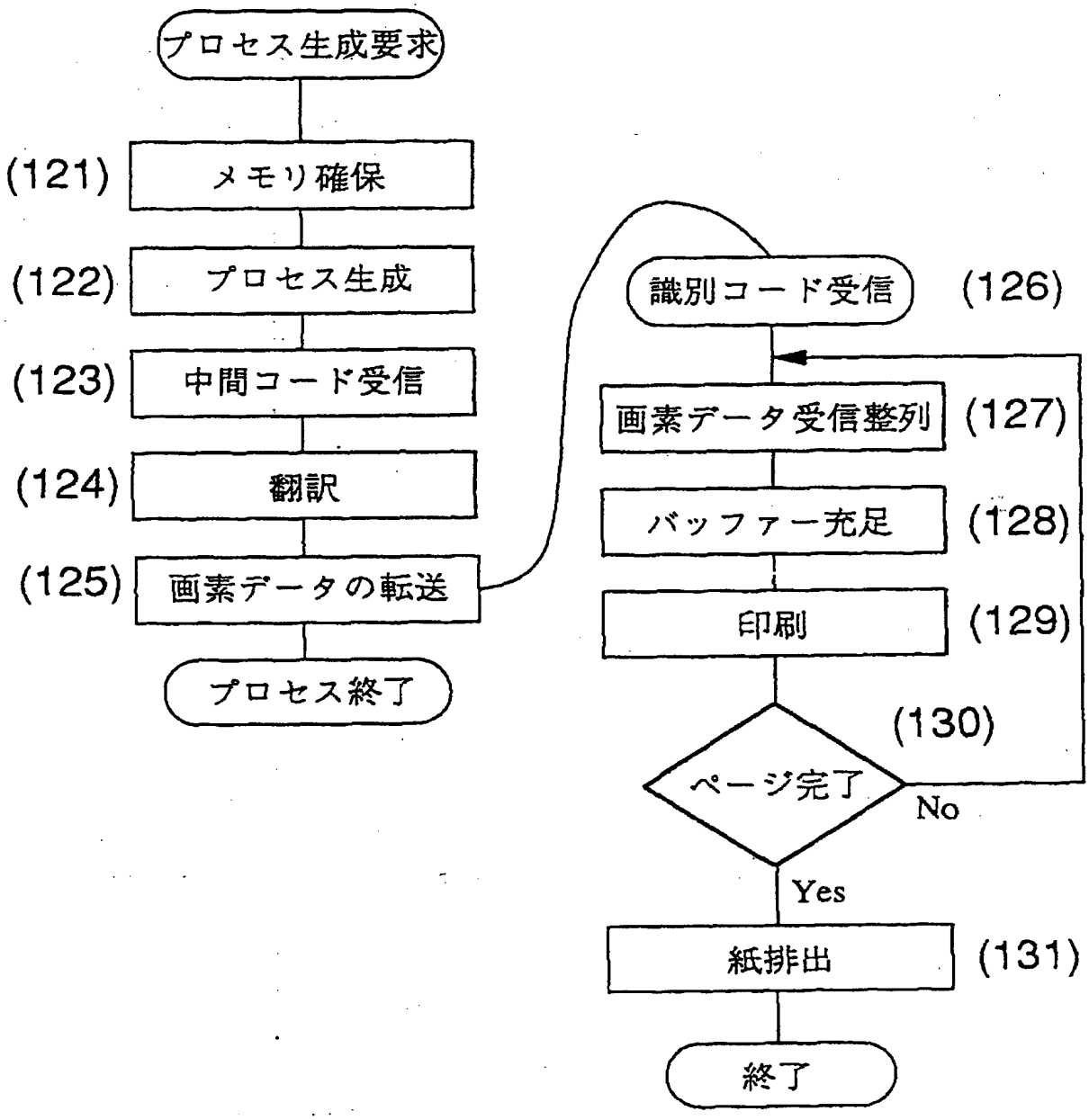
【第３１図】


6a
6b
6c
18
106
106
106
4
212
4
212
4
212
213
213
213
7
108
107
グループ検出
302
301
303

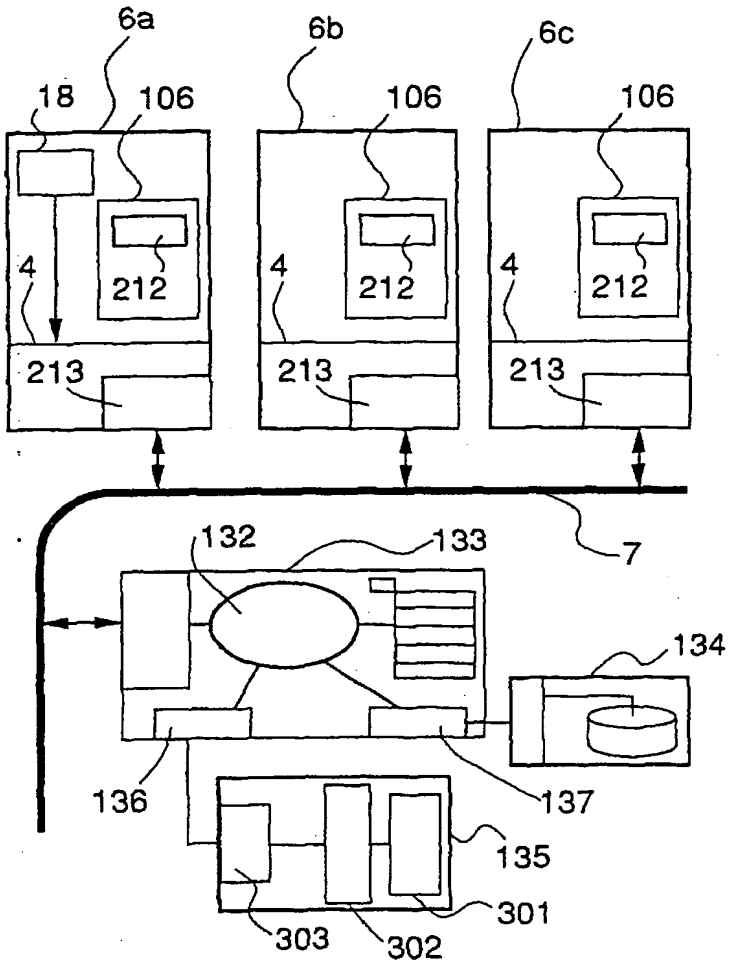
【第３２図】


印刷要求発生
(109)
ページ開く
(110)
文字／図形の記述
(111)
ページを閉じる
(112)
印刷ファイル生成
(113)
印刷装置へ転送
終了
識別コード受信
(114)
ファイル受信
(115)
中間コードへの翻訳
(116)
図形グループ検出
(117)
グループ分割
(118)
プロセス生成要求
(119)
中間コード転送
(120)
印刷データ待ち

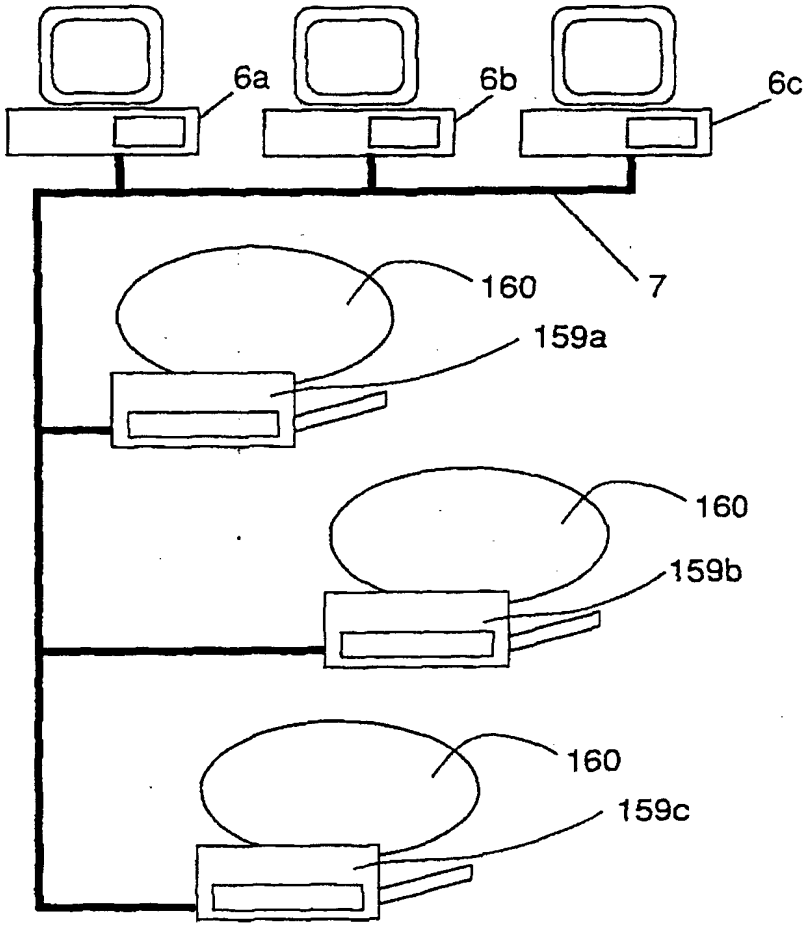
【第３３図】


プロセス生成要求
(121)
メモリ確保
(122)
プロセス生成
(123)
中間コード受信
(124)
翻訳
(125)
画素データの転送
プロセス終了
識別コード受信
(126)
画素データ受信整列
(127)
バッファー充足
(128)
印刷
(129)
(130)
ページ完了
No
Yes
紙排出
(131)
終了

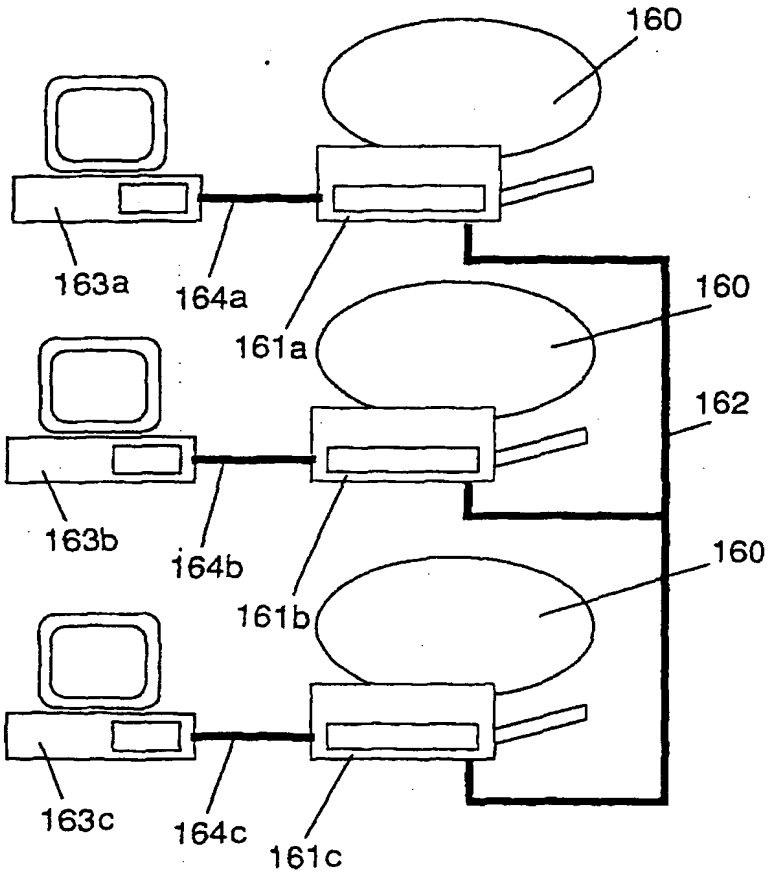
【第３４図】


6a
6b
6c
18
106
106
106
4
212
4
212
4
212
213
213
213
7
132
133
134
136
137
135
301
303
302

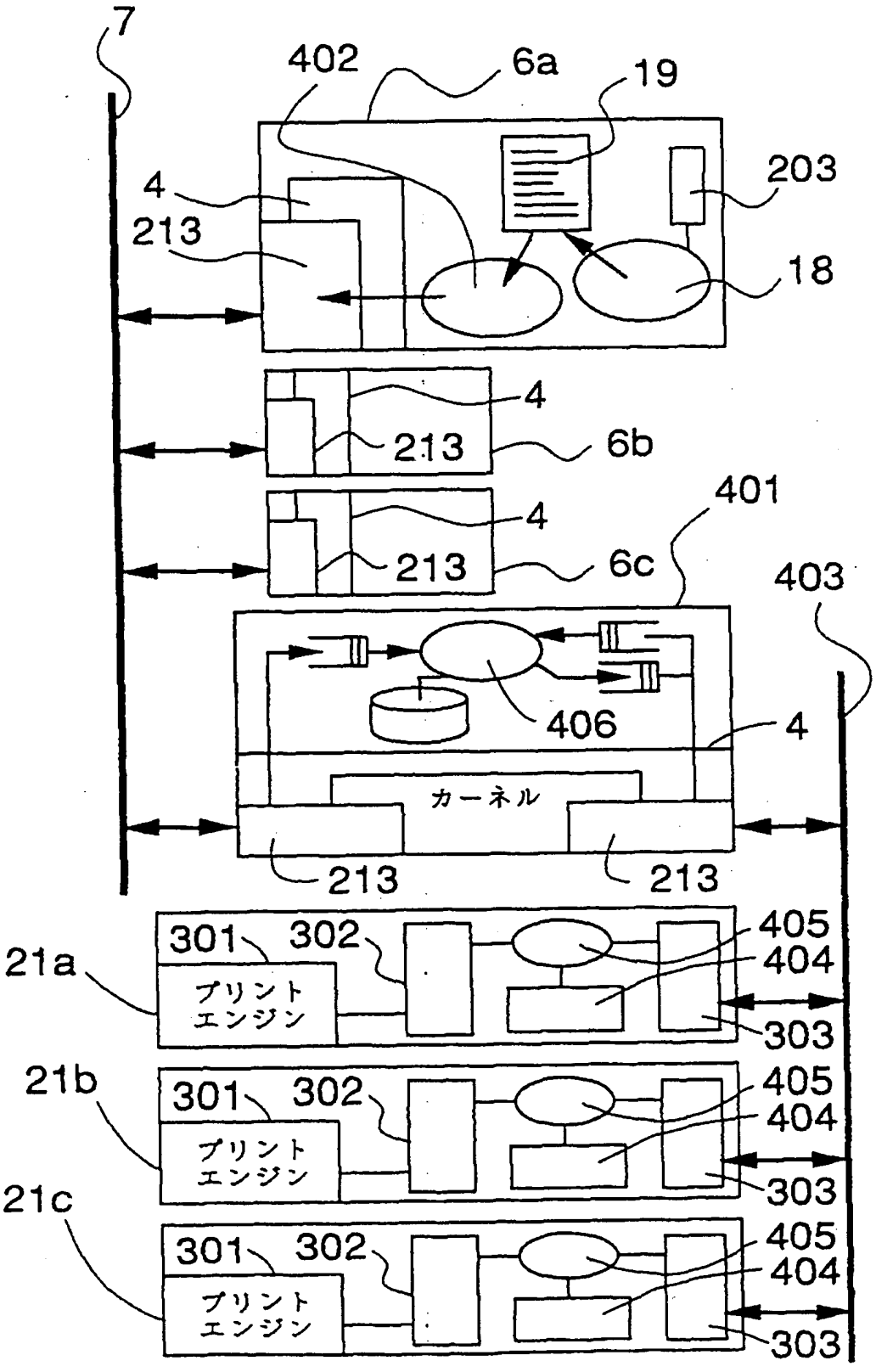
【第３５図】



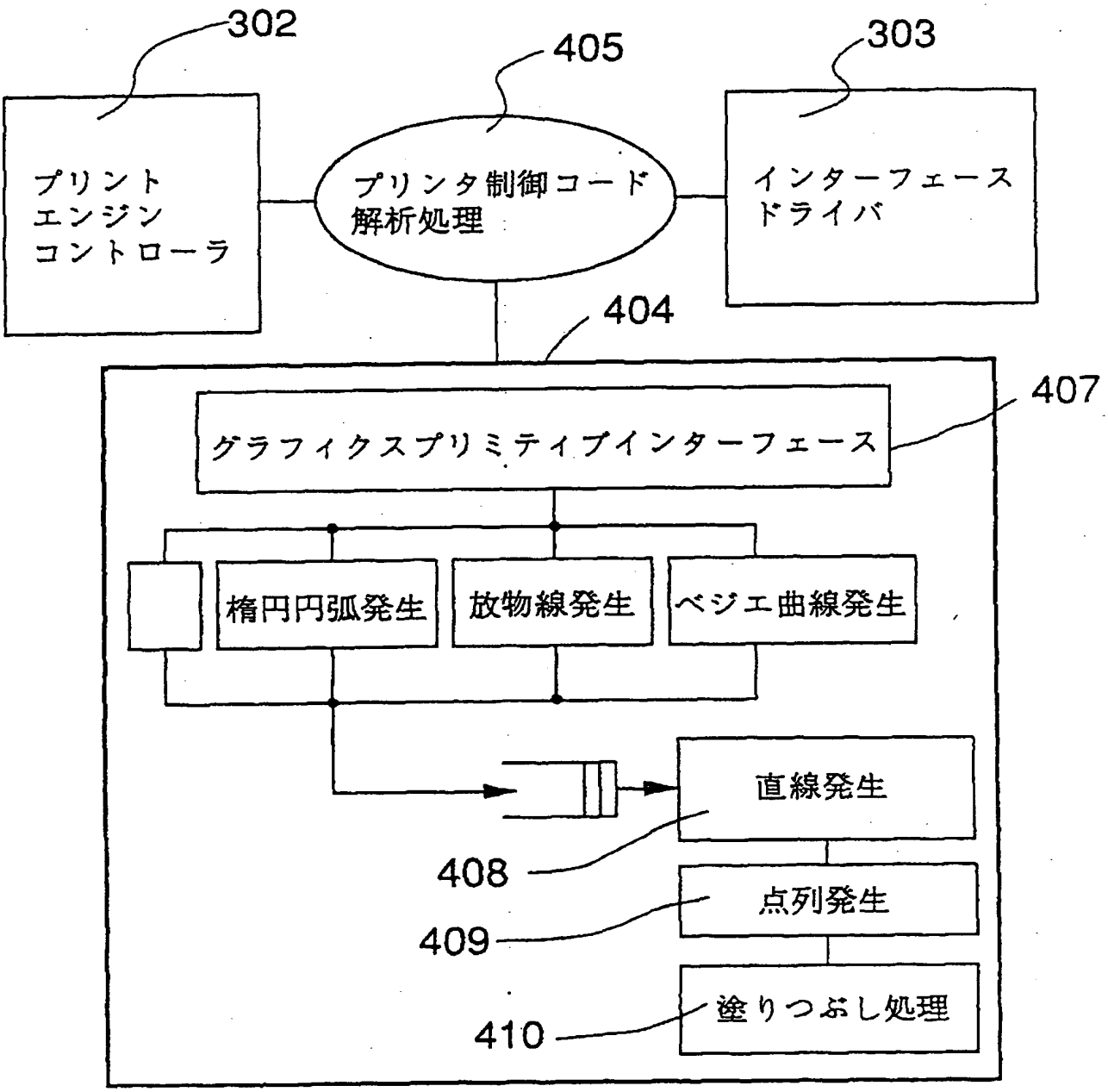
【第３６図】



【第３７図】


7
402
6a
19
203
4
213
18
4
213
6b
401
4
213
6c
403
406
4
カーネル
213
213
21a
301
302
プリント
エンジン
405
404
303
21b
301
302
プリント
エンジン
405
404
303
21c
301
302
プリント
エンジン
405
404
303

【第３８図】


302
405
303
プリント
エンジン
コントローラ
プリンタ制御コード
解析処理
インターフェース
ドライバ
404
407
グラフィクスプリミティブインターフェース
楕円円弧発生
放物線発生
ベジエ曲線発生
直線発生
408
点列発生
409
塗りつぶし処理
410